SORBONNE UNIVERSITÉ

EURECOM
Sophia Antipolis

Employing Transformers and Humans for Textual-Claim Verification

Dissertation

submitted to

Sorbonne Université

in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Author:

Mohammed SAEED

Scheduled for defense on the 7th of November 2022, before a committee composed of:

Reviewers

Prof.	**Tanmoy CHAKRABORTY**	IIT Delhi, India
Prof.	**Andreas VLACHOS**	University of Cambridge, England

Examiners

Prof.	**Fabian SUCHANEK**	Télécom Paris, France
Prof.	**Maria ZULUAGA**	EURECOM, France

Thesis Advisor

Prof.	**Paolo PAPOTTI**	EURECOM, France

Utilisation de Transformateurs et d'Humains pour la Vérification de Revendications Textuelles

Thèse

soumise à

Sorbonne Université

pour l'obtention du Grade de Docteur

présentée par:

Mohammed SAEED

Soutenance de thèse prévue le 07 Novembre 2022 devant le jury composé de:

Rapporteurs

Prof.	**Tanmoy CHAKRABORTY**	IIT Delhi, Inde
Prof.	**Andreas VLACHOS**	Université de Cambridge, Angleterre

Examinateurs

Prof.	**Fabian SUCHANEK**	Télécom Paris, France
Prof.	**Maria ZULUAGA**	EURECOM , France

Directeur du thèse

Prof.	**Paolo PAPOTTI**	EURECOM, France

To my beloved family whose immense love makes me flourish

"I put my heart and my soul into my work, and have lost my mind in the process."
— Van Gogh

"Happy to see you going out for a change."
— Mom

Abstract

Throughout the last years, there has been a surge in false news spreading across the public, ranging from biased political campaigns, to the propagation of misinformed coronavirus claims. This has especially seen more interest as advances in deep learning made it feasible to produce sound text steered towards disinformation. Despite efforts made in alleviating "fake news", from the identification of check-worthy claims to the extraction of evidence for claim verification, there remains a lot of ordeals when trying to build automated fact-checking systems, including the four we discuss in this thesis. First, it is not clear how to bridge the gap between input textual claims, which are to be verified, and structured data that is to be used for claim verification. We take a step in this direction by introducing Scrutinizer, a data-driven fact-checking system that translates textual claims to SQL queries, with the aid of a human-machine interaction component. We then apply the system to two use cases: the energy and health domains. Second, many data-driven solutions for fact-checking rely on pre-trained language models (PLMs), where one application is for the verification of a claim. While PLMs exhibit some form of logical and reasoning capabilities, they still lack in basic logical notions such as negation and symmetry. One step in this direction is RuleBERT, a PLM that is fine-tuned on data coming from logical rules. We discuss how to generate the data and incorporate weights of soft rules during training, showing more consistency in terms of negation and symmetry. Third, PLMs store vast information; a key resource in fact-checking applications. Still, it is not clear how to efficiently access them. Several works try to address this limitation by searching for optimal prompts or relying on external data, but they do not put emphasis on the expected type of the output. For this, we propose Type Embeddings (TEs), additional input embeddings that encode the desired output type when querying PLMs. We discuss how to compute a TE, and provide several methods for analysis. We then show a boost in performance for the LAMA dataset and promising results for text detoxification. Finally, while PLMs are a viable approach for computational fact-checking, there is no escape from including humans-in-the-loop. In this direction, we analyze the Birdwatch program, a community-driven approach to fact-checking tweets. We investigate how users choose claims, what sources of information they use, and whether their assessment is viable by comparing it to that of expert fact-checkers. This study allows us to compare and contrast the alternative methods and envision collaborative systems where computational methods work hand in hand with humans. All in all, the work in this thesis aims at a better understanding of how machines and humans could aid in reinforcing and scaling manual fact-checking.

Abrégé

Ces dernières années, on a assisté à une recrudescence de la diffusion de fausses nouvelles dans le public, qu'il s'agisse de campagnes politiques biaisées ou de la propagation d'affirmations de coronavirus mal informés. Ce phénomène a suscité un intérêt accru depuis que les progrès de l'apprentissage profond ont rendu possible la production de textes sonores orientés vers la désinformation. Malgré les efforts déployés pour atténuer les "fake news", de l'identification des affirmations dignes d'être vérifiées à l'extraction de preuves pour soutenir la vérification des affirmations, il reste beaucoup d'épreuves à surmonter lorsqu'on essaie de construire des systèmes de vérification automatique des faits, y compris les quatre que nous abordons dans cette thèse. Tout d'abord, il n'est pas clair comment combler le fossé entre les revendications textuelles d'entrée, qui doivent être vérifiées, et les données structurées qui doivent être utilisées pour la vérification des revendications. Nous faisons un pas dans cette direction en présentant Scrutinizer, un système de vérification des faits piloté par les données qui traduit les affirmations textuelles en requêtes SQL, avec l'aide d'un composant d'interaction humain-machine. Nous appliquons ensuite le système à deux cas d'utilisation, notamment dans les domaines de l'énergie et de la santé. Deuxièmement, de nombreuses solutions de vérification des faits basées sur les données reposent sur des modèles de langage pré-entraînés (PLM), dont l'une des applications est la vérification d'une déclaration. Si les PLM présentent une certaine forme de capacités logiques et de raisonnement, ils ne maîtrisent toujours pas les notions logiques de base telles que la négation et la symétrie. Un pas dans cette direction est RuleBERT, un PLM qui a été développé sur des données provenant de règles logiques. Nous discutons de la manière de générer les données et d'incorporer les poids des règles logiques pendant l'apprentissage, ce qui montre une plus grande cohérence en termes de négation et de symétrie. Troisièmement, les PLM stockent de vastes informations qui constituent une ressource clé dans les applications de vérification des faits. Pourtant, la manière d'y accéder efficacement n'est pas claire. Plusieurs travaux tentent de remédier à cette limitation en recherchant des prompts optimales ou en s'appuyant sur des données externes, mais ils ne mettent pas l'accent sur le type attendu de la sortie. C'est pourquoi nous proposons les Embeddings de Type (TEs), des embeddings d'entrée supplémentaires qui encodent le type de sortie souhaité lors de l'interrogation du PLM. Nous expliquons comment calculer un TE et fournissons plusieurs méthodes d'analyse. Nous montrons ensuite une augmentation des performances pour le jeu de données LAMA et des résultats prometteurs pour la détoxification de textes. Enfin, si les PLM constituent une approche viable pour le

fact-checking computationnel, il n'est pas possible d'échapper à l'inclusion des humains dans la boucle. Dans cette direction, nous analysons le programme BIRDWATCH, une approche communautaire de vérification des faits dans les tweets. Nous étudions comment les utilisateurs choisissent les affirmations, quelles sources d'information ils utilisent, et si leur évaluation est viable en la comparant à celle d'experts en vérification des faits. Cette étude nous permet de comparer et de contraster les méthodes alternatives et d'envisager des systèmes collaboratifs où les méthodes informatiques travaillent main dans la main avec les humains. Dans l'ensemble, les travaux de cette thèse visent à mieux comprendre comment les machines et les humains pourraient contribuer à renforcer et à étendre la vérification manuelle des faits.

Acknowledgements

Doing a PhD was not in my visionary growing up; until I received this email one-day stating that there are summer internship positions with a professor who just joined Eurecom. I decided to apply, but was later hesitant to complete the process. Luckily, I did, and met Paolo, who later on turned to be my PhD advisor. I have learned a lot in this PhD, and I thank Paolo for the time, care, and expertise he shared with me which shaped the researcher I am today. I will forever be grateful.

I have had the pleasure to be involved in amazing intercontinental collaborations with Immanuel, Preslav, and Gianluca, whom I leaned immensely from.

This journey would not have been without the amazing crew at Eurecom. My colleague Naser was always bringing a smile to my face with his hilarious stories. Ismail was always making me cheerful with his jolly and humorous spirit. Riccardo was an amazing office-mate, and I still remember the vast and diverse topics of conversations we used to indulge in. Finally, a shoutout to Jean-Flavien, Youssef, Kensuke, Youri, and Youssra for the fun time we had during my last months here.

My journey was exceptional thanks to all the amazing friends I met here. Ahmad, thank you for being there for me during the thick and thin, and encouraging me to push more. I will forever cherish our friendship. Oussama, thank you for all the amazing time we had here before leaving. Adib, thank you for all the wonderful adventures in the region. Haydar, I am beholden to you, as you have been and will always be a close friend. Thank you to Dima and Hind, for all the great times we spent here, making me feel as if I were back in Lebanon.

Finally, I wouldn't be here without the unconditional love and support I had from my family: mama and baba, Tahani and Adel, Nour, Rania, and Teya. They have always encouraged me to pursue my passions and will forever be my bedrock, especially my role-model mother. I am happy to have had the chance to visit Boston and meet my aunt and cousins, with whom I share exceptional experiences. My upbringing was phenomenal with all the love and care (and food), I had from my maternal grandparents who both saw the beginning of this thesis, but my grandfather did not see the end. Both your love and support will always carry me.

Contents

List of Figures

List of Tables

1 Introduction

From politicians to advertisers, from advocacy groups to enterprises — everyone who seeks to persuade others has an incentive to distort, exaggerate or obfuscate the facts. The topic of *fake news* has experienced a substantial resurgence of interest in our society. This is not surprising as the number of fact-checking organizations nearly doubled from 186 active organizations in 2016, to 391 active fact-checking projects in 2021 across 105 countries (Stencel et al., 2022). In addition to that, there was an increase of engagement with fake news content in social media platforms such as Facebook (Allcott et al., 2019) and Twitter (Hindman and Barash, 2018), which is exacerbated by the millions of fake-news-spreading bots (Ferrara et al., 2016). However, fake news is not something new; in fact, fake news has been spreading ever since the printing press was created back in the 15^{th} century. One particular instance is that of the 'The Great Moon Hoax', where the *New York Sun* newspaper published a series of illustrated articles in 1835 about mythical creatures living on the surface of the moon. Such made-up stories were viral until the writers of the stories declared them to be satire (Posetti and Matthews, 2018). Indeed, nowadays, we have started to catch a glimpse of the dire effects fake news could have on several angles including our society, economy, and personal health. For example, during the US presidential elections of 2016, articles about the Pope endorsing Donald Trump and Hilary Clinton's ISIS email getting leaked had more engagement than mainstream news (Armstrong, 2016); where the most engaged fake articles tended to favor Donald Trump over Hillary Clinton. Some work suggested such influence actually steered the elections in favor of Trump (Dewey, 2016; Parkinson, 2016). On the economy level, fake news always had its shares. A prime example of this is when a group of people who planned, in a virtual chat room, to promote a certain crypto coin through a fake 'John McAfee' Twitter account (Shome, 2017). After sending out a tweet from the account declaring that the coin was the coin of the day, its value skyrocketed, and the chat group were able to identify the best times to buy and sell the coin, before the value returned to its original cost later. In reality, a study estimated the cost of online fake news on the global economy to be at least $78 billion annually (CHEQ, 2019). Last but not least, fake news has drastically affected our health with the flood of false coronavirus

news; especially, about those promoting fake coronavirus cures, where at least 800 people have died in the first 3 months of 2020 by drinking methanol or alcohol-based cleaning products, believing it was a cure for the virus (Coleman, 2020). Indeed, it is not a surprise that massive digital misinformation has been designated as a major technological and geopolitical risk by the 2013 report of the World Economic Forum (Howell, 2013), and an 'infodemic' by the World Health Organization causing confusion and risk-taking behaviors that can harm health (Organization, 2022).

1.1 Fake News

1.1.1 A Brief Definition

While Collins Dictionary named it word of the year in 2017, there is no agreed on definition of fake news (Shu et al., 2017; Sharma et al., 2019). Fake news has generally been defined as "news articles that are intentionally and verifiably false, and could mislead readers" (Shu et al., 2017; Allcott and Gentzkow, 2017). Another definition includes news fabrications, satires, and hoaxes (Balmas, 2014; Rubin et al., 2016). However, such definitions are narrow, restricted either by the type of information or the intent of deception, and do not capture the broader scope of the term based on its current usage. Therefore, we define fake news similarly to (Sharma et al., 2019) as follows:

Definition 1. *A news article or message published and propagated through media, carrying false information regardless the means and motives behind it.*

Such definition allows covering a broader aspect of fake news, more aligned with its usage, and regardless of the intent of the user.

1.1.2 Why is Fake News a Matter of Concern?

It is hardly surprising that fake news has had grave consequences on several aspects including political (Parkinson, 2016), economical (CHEQ, 2019), and physiological (Coleman, 2020). Stanford researchers, conducting a study, were 'shocked' by how young Americans could not differentiate between fake and real news (Stanford History Education Group, 2016). As a large majority of people rely on social media to get their news, this has become exceedingly worrying (Shearer, 2021). Such worry is validated by a recent Harvard study on participants in 142 countries, showing that the majority of regular internet users globally (58.5%) are concerned about misinformation (Knuttila et al., 2022).
While fake news has a plethora of worrying concerns, a possible explanation of its ramifications could be accounted for by comparing with common challenges observed in modern data management (Abiteboul et al., 2015):

- **The Volume of Fake News.** It is within anyone's capability to spread any sort of news on the web, as some form of proper verification is either lacking on some social medias, or can not cope up with the amount of fake news. Some social media

Tweet	`Pregnant women, please don't take this vaccine.` `https://t.co/4KKlnMIbl7`
User Fact-check	`Updated CDC guidance, and newly accepted and reviewed medical research, has stated there are no safety concerns for pregnant women to be vaccinated against COVID-19.` *(links omitted for brevity)*
Expert Fact-Check	`The vaccine is safe for pregnant women or women planning on becoming pregnant within a few months of taking the vaccine.`

Table 1.1: A tweet promoting fake news being debunked by a Twitter user and an expert fact-checker.

platforms have intervened by cooperating with third party organizations (Meta, 2016) and adopting a community-based approach to misinformation (Coleman, 2021a). However, there does still exist numerous pages on the web that spread fake news (Allen et al., 2020).

- **The Velocity of Fake News.** Fake news can be generated and spread rapidly. Regarding the generation aspect, with recent advances in natural language generation (NLG) models (Otter et al., 2021; Li et al., 2021b) and computation power, it has become a sinecure to generate a deluge of textual claims that are coherent without any guarantees on their factual correctness. Regarding the spread of fake news, a study on Twitter showed that tweets containing false claims reached six time more people than regular news (Langin, 2018), resulting in more engagement with users on social media, and possibly more severe socio-economical effects and health risks.

- **The Variety of Fake News.** Not all fake news is created equally; in fact, it can take many forms such as *satire* (manipulated news intended for the use of humor, irony, exaggeration, or ridicule to expose and criticize people's stupidity or vices), *clickbaits* (intentionally misleading headlines or titles for the purpose of encouraging a user to click through and visit a certain website), and *conspiracy theories* (manipulated information spread deliberately to affect political or social institutions) (Carrasco-Farré, 2022). Deliberately faking news with the intent to harm is often termed as *Disinformation*, while that due to unintentional mistakes is termed as *Misinformation*. Fake news can also span various topics (Figure 1.3), and occur in claims requiring some form of numerical computation to be verified, or logical claims needing some form of facts and logical rules for assessing truthfulness (Figure 1.1).

- **The Veracity of Fake News.** Verifying an input claim is by no means a trivial

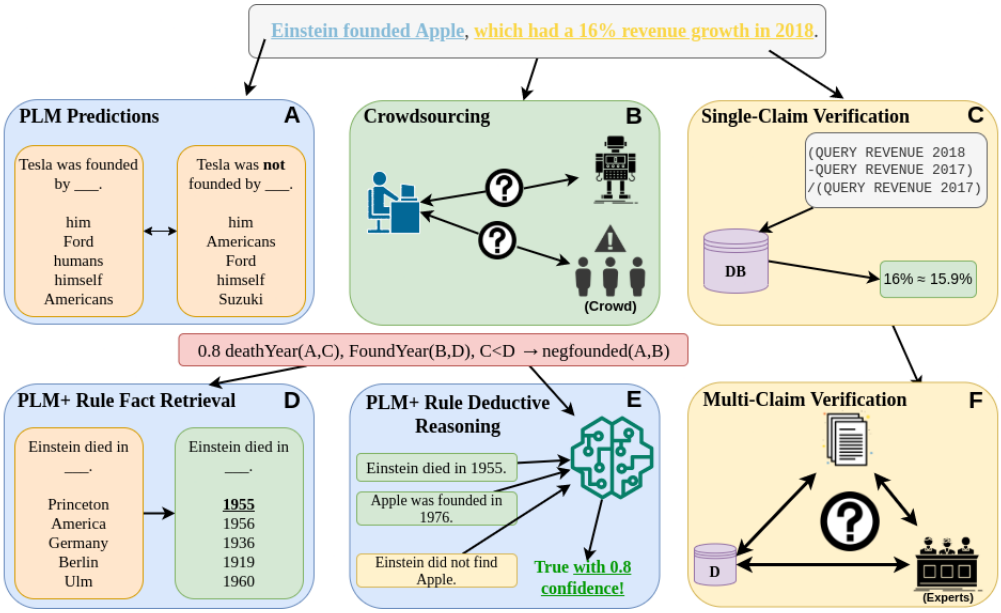

Figure 1.1: Some problems encountered during claim verification.

task, as there is still no agreed upon methodology for properly assessing a claim. As different places have different standards of journalism, let alone varying levels of press freedom, it is no surprise that fact-checking methodologies adapt to their circumstances. While the International Fact-Checking Network published a code of principles for fact-checkers[1], such principles are far from being adopted by fact-checkers and fact-checking organizations. Moreover, verification of a claim cannot be modeled as a yes/no problem, as there are multiple shades of truthfulness of claims. PolitiFact, one of the renowned fact-checking organizations, defines the truthfulness of a statement according to a scale of six ratings[2], while Snopes on the other hand defines 17 different ratings[3]. It is therefore strenuous to discretize the degrees of truthfulness of a claim and uniform them across the various rating schemes. The MisInfoMe dataset tries to normalize labels coming from various sources under 5 ratings (Mensio and Alani, 2019). However, verifying the authenticity of fake news does not need to be done by trained professionals only. Recently with the Twitter's Birdwatch program, users, with no formal fact-checking training, can also participate in identifying misleading information (Table 1.1, Figure 1.3).

[1] http://www.poynter.org/ifcn-fact-checkers-code-of-principles/

[2] https://www.politifact.com/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/

[3] https://www.snopes.com/fact-check-ratings/

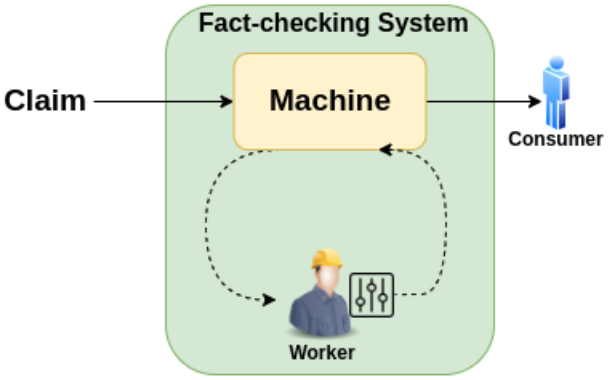

Figure 1.2: Humans can be included in a fact-checking system as (i) workers who interact with the system and update its performance or (ii) consumers who digest the system output. Workers can be experts or humans without any formal training. Consumers are humans who do not interact with the system and witness the output results only.

1.2 Fact-Checking

1.2.1 Journalistic Fact-Checking

In the pre-automation period, checking for fake news was performed by trained journalists. These journalists were employed to proofread and verify claims made in written or spoken language. This type of fact-checking verifies the solidity of the stated claims, and serves as an overall round of quality control for a news outlet's content *before publication*; acting as a core part of journalistic work (Cazalens et al., 2018; Mantzarlis, 2018). Such fact-checking procedure is a vital component in the news reporting process. In their book, "The Elements of Journalism", Tom Rosenstiel and Bill Kovach wrote that "The essence of journalism is a discipline of verification.". Both fact-checking and verification are complementary rather than conflictual, as the former is used in the latter (Mantzarlis, 2015). In this thesis, we use both terms interchangeably.

In this thesis, we tackle the second type of fact-checking, which happens not before something is published but rather after a claim becomes public. This form of *a posteriori* fact-checking is often performed by specialists in active NGOs such as PolitiFact, FullFact, and Les Décodeurs. The fact-checking work concentrates primarily, but is not limited to, political ads, campaign speeches and party manifestos (Mantzarlis, 2018). We define a *claim* as a snippet of text for which there is interest in knowing whether it is valid or not.

1.2.2 Why Manual Fact-Checking is not Enough?

With the struggles of manual fact-checkers to keep up with the ever-increasing surge of fake news (Horwitz, 2020), we are facing a widening gap between the growing amount of data on one hand, and the shrinking body of trained journalists on the other (Jennings, 2020). Alas, relying solely on manual fact-checkers for the fight is not enough. One

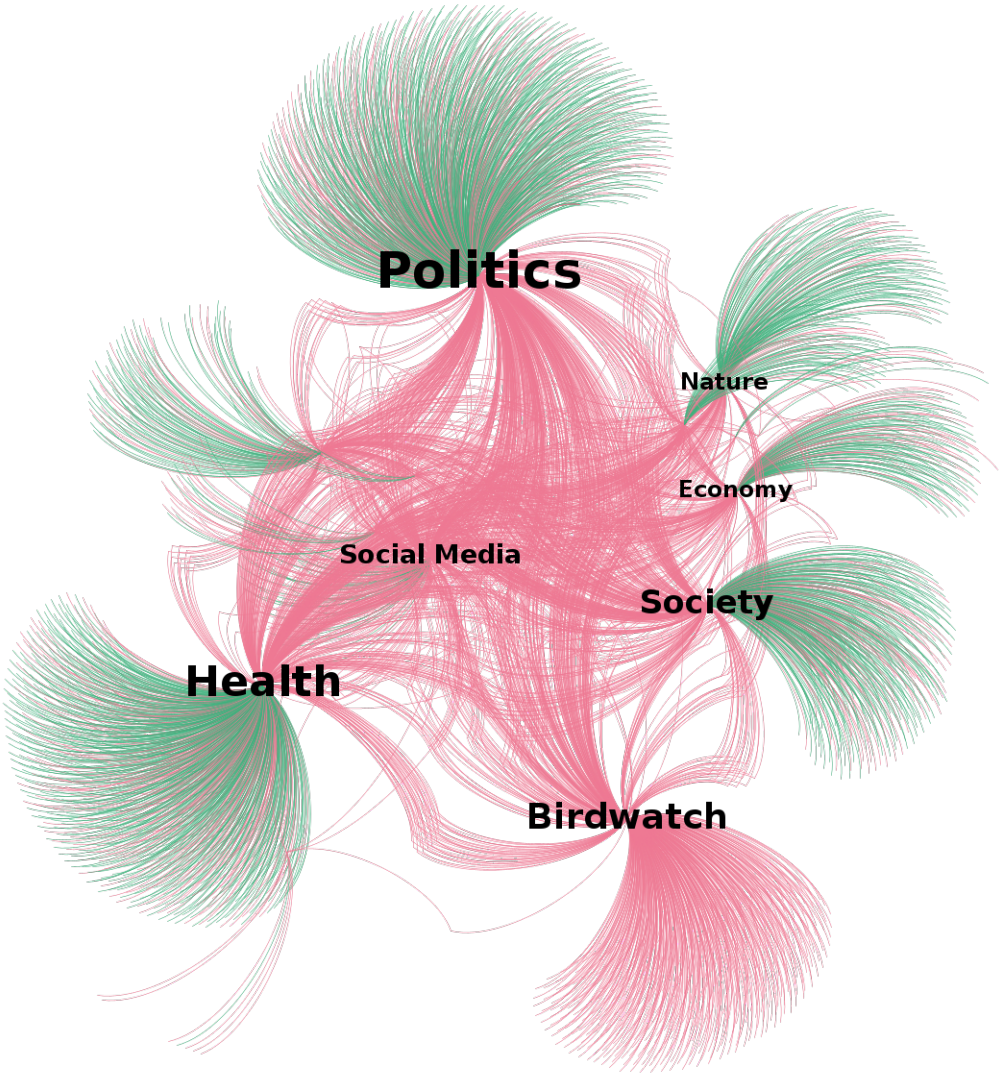

Figure 1.3: A graph showing the tweets from Birdwatch data matched with ClaimReview fact-checks, clustered by topic. Pink lines indicate tweets that have been checked by Birdwatch users only, while the green ones show tweets that have been matched to fact-checks performed by experts (The process of matching between tweets and expert fact-checks is explained in Section 6.3.3). The tweets have been grouped by topic, where larger fonts indicate topics with high relevance. Indeed, topics such as *Politics* and *Health* are more dominant, and seem to be check-worthy by both Birdwatch users and expert fact-checkers. As fact-checks done by experts are costly, the majority of tweets are only fact-checked by Birdwatch users who are not formally trained for fact-checking claims.

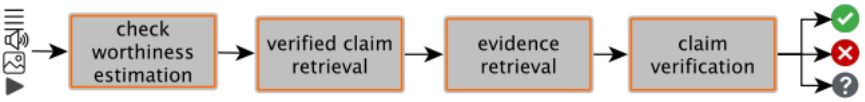

Figure 1.4: A Fact-Checking pipeline (taken from (Nakov et al., 2021a)).

possible direction to narrow this gap is through the efficient use of computing algorithms and resources. Computing methods would, ideally, attempt to imitate a fact-checker as much as possible. This would not only ease some manual aspects of the fact-checking process, but could help, to a certain extent, with the enormous volume of fake news produced every day. For such reason, it is not a surprise that *Computational Fact-Checking* has sparked interest across academic labs and industries, and has emerged as an aspiring research field among academics (Vlachos and Riedel, 2014; Hassan et al., 2015).

1.2.3 Computational Fact-Checking to the Rescue

There have been calls to automate the fact-checking process. (Vlachos and Riedel, 2014) envision the task as providing tools to journalists that would allow greater ease in the process due to advances in computational domains such as natural language processing, databases and information retrieval. They also released the first dataset by crawling statements from PolitiFact and Channel 4. This was then followed by several fact-checking-related tasks, such as the *CLEF CheckThat! lab* (Elsayed et al., 2019; Nakov et al., 2021b) and the Fact Extraction and Verification shared task (FEVER) with its derivatives (Thorne et al., 2018a; Christodoulopoulos et al., 2020; Aly et al., 2021). On the other hand, (Hassan et al., 2015) ideate a computer-based fact-checking system that is fully automated, instant, accurate, and accountable.

The process of fact-checking is usually envisioned with a certain pipeline. A FullFact report associates automating fact-checking with the automation of a four-stage pipeline consisting of (i) monitoring claims, (ii) spotting claims, (iii) checking claims, and (iv) publishing the fact-checking results (Babakar and Moy, 2016). Another typical pipeline is shown in Figure 1.4. It is composed of four main stages: (i) detecting claims that are check-worthy, (ii) retrieving claims that have been previously fact-checked, (iii) gathering evidence that is vital for (iv) claim verification where a final verdict is expected. We will adopt this pipeline in the thesis.

Note that any fact-checking pipeline ideally can digest any form of manipulated input, including images and videos, and not necessarily textual data. As for the output of such pipelines, the approaches can be divided into explainable and non-explainable (Nakov et al., 2021a). Explainable approaches are more relevant for fact-checkers, as they provide evidence needed to explain the system's decision. In this thesis, we lean more towards explainable methods to fact-checking with only textual claims as input, as they are the main focus of fact-checking organizations.

1.3 Challenges for Computational Fact-Checking

Despite the numerous fact-checking systems currently present (Guo et al., 2022b), we are yet far from fully automating the fact-checking pipeline (Figure 1.4) due to several ordeals. We tackle 4 of them in this thesis:

1. Despite the plethora of fact-checking systems built, incorporating structured data, specifically relational tables, into the fact-checking procedure is still an open-problem. This is of valuable interest, as relational tables are one of the most prominent data storage formats. While there exists many systems that attempt to integrate tabular information (Borisov et al., 2021; Gorishniy et al., 2021), the majority of such systems has been applied on well-curated datasets, disregarding the challenge posed by domain-specific claims that require costly annotations of domain experts. Given the statistical claim in Figure 1.1, verifying such a claim would require identifying key cells in a database **DB** (shown in **C**), followed by the use of some mathematical formula to compute revenue growth; such formulas can even be more complex such as the compound annual growth rate[4]. As more and more data is abundantly stored in relational tables[5], verification of textual claims renders more cumbersome, as the gap between unstructured textual data and structured relational tables needs to be addressed. With the limitations offered by machine-learning models, including humans in the fact-checking procedure is inevitable. Figure 1.2 shows a high-level overview of a fact-checking system with humans. Humans can be on the consuming end of the system, digesting the output; or part of the fact-checking system itself, where they interact with machines (e.g., through a machine-learning model (Maadi et al., 2021) or a scheduling algorithm (Difallah et al., 2016)) steering the system's behavior. However, in practical settings, multiple claims exist where manual verification becomes tedious and intractable; especially when claims relate to a specific domain that needs expert fact-checkers, thus making the cost of verification exorbitant. How does one go about verifying numerous claims without relying excessively on expert fact-checkers for manual labeling, while coping with the imperfections of machine learning models? An approach that utilizes machine learning models for scalability while soliciting feedback from experts in an optimal fashion would be needed (Figure 1.1(F)) for such an ordeal.

2. Despite that recent work utilizes pre-trained language models (PLMs) for claim verification, PLMs still lack in terms of logical reasoning, as they cannot perform deductive reasoning using logical rules (Figure 1.1(E)). They are also severely inconsistent. Figure 1.1(A) shows how a sentence and its negated form have similar outputs from a PLM. While several work tries to address this ordeal on generic neural networks, a few approaches tackle this for transformer-based PLMs. Augmenting PLMs with logic is worthwhile, as, aside from their ground-breaking performances on several language tasks, incorporating reasoning in them would

[4] https://en.wikipedia.org/wiki/Compound_annual_growth_rate
[5] https://db-engines.com/en/ranking_trend

allow PLMs to enhance in logic-related tasks such as answering questions that need some form of reasoning (Dua et al., 2019; Clark et al., 2019a; Liu et al., 2020a). Despite that PLMs incur some form of logical notion (Talmor et al., 2020a), they do suffer from inconsistency (Elazar et al., 2021) in terms of negation (Kassner and Schütze, 2020) and symmetry (Ribeiro et al., 2020). However, even for some line of work that augments PLMs with logical rules (Clark et al., 2020; Saha et al., 2020; Tafjord et al., 2021), the associated rules are simplistic as they only consider hard rules with single-variable predicates. This is not sufficient for real-world first-order logic requiring predicates with multiple variables and rules with an accompanying weight to measure their validity (Russell and Norvig, 2010; Fierens et al., 2014). How can a PLM "reason" with facts, rules, and hypothesis in natural language with probabilistic outputs (Figure 1.1(E))?

3. As PLMs store vast factual and common-sense information, it is no surprise that they are used for fact retrieval in domains such as fact-checking (Lee et al., 2020) and rule mining (Cui and Chen, 2021). Obtaining the capital of Chile is merely but a query as `The capital of Chile is` _____, where the PLM must fill in the blanks with the correct answer `Santiago`. However, one cannot always expect to arrive at their desired output. Figure 1.1(D) shows how PLMs predict location outputs, contrary to the intent of the user to retrieve birth years instead. Thus, one caveat of querying PLMs is the inability to control the output type. Being able to direct the PLM when querying to match a desired criterion is required for better retrieval, and eventually better performance on related tasks. While approaches that search for optimal prompts through means of learning algorithms, or rely on external resources to enrich the input context improve performance, they do not take the desired output type into account on one hand, and require training the model or relying on external resources on the other. How can we enforce the desired type, knowing that PLMs encode latent concepts such as city and year (Dalvi et al., 2022) (Figure 1.1(D))?

4. As automating the entire fact-checking pipeline is currently infeasible, including humans in the pipeline is inevitable. As expert fact-checkers are scarce, a certain methodology remedies this by relying on a larger number of amateur humans to perform fact-checking (Figure 1.3); or what is known as the 'wisdom of the crowd'. While a series of works utilize such crowd-sourcing approach to fact-checking (Roitero et al., 2020a, 2021), it has been only done for controlled environments that undermine communication between the crowd, and there is no clear vision of how such an approach would behave in real uncontrolled settings. An analysis in such settings is needed as to better understand what each user in the crowd has to offer *per se*, compared to one another, and compared to expert fact-checkers. Is such an approach effective? And how does it compare to experts and computational methods (Figure 1.1(B))?

1.3.1 Contributions

The remainder of the thesis is organized as follows:

- Chapter 2 comprises the background knowledge required for the rest of the chapters of the thesis.
- Chapter 3 presents a fact-checking system, SCRUTINIZER, that verifies statistical claims in natural language by exploiting the synergy between humans and algorithms. The system is based on two components: a claim translation component and a question planning component. The former is responsible for automated translation from text to query elements (such as datasets and attributes) needed to fact-check the claim, done through the use of trained classifiers. However, as data is not always available, the system needs to solicit feedback from experts for labeling the data. This is where the latter component comes into play. The question planning component interacts with human domain experts in a manner that optimizes verification tasks for maximal benefit. This is done by modeling claim selection as an integer linear programming problem that allows to select the most optimal claims to be labeled. We apply SCRUTINIZER to two use-cases: one domain-specific related to energy, and the other to coronavirus.

  > Chapter 3 comprises the evidence retrieval and claim verification stages of Figure 1.4. The novelty lies in a Text2SQL fact-checking system applied to real world data, where users can be involved in the interaction with the machine, or as consumers of the system's output.

- Chapter 4 explores how to emulate reasoning with PLMs using first-order soft logic rules. We extend previous work by (i) harnessing PLMs with logical rules containing binary predicates, as opposed to unary predicates, and (ii) incorporating soft rules during training. For (i), we develop a data generation algorithm that encompasses logical aspects such as symmetry, which is crucial when dealing with binary predicates, and negation. For (ii), we propose to modify the objective function to integrate the weights of rules. To do so, we go by weighting data-points by the rule confidence and creating their virtual counter-part before training. We test our system RULEBERT on single rules of various confidences, multiple rules including rules with conflicting conclusions, and chained rules. We finally test RULEBERT on several external datasets showing improvements in deductive reasoning and in logical notions such as negation and symmetry. We also release this novel dataset, comprising 3.2M examples derived from 161 logical rules.

  > Chapter 4 encompasses the claim verification stage of Figure 1.4. The contribution lies in studying how well first-order soft logical rules incorporate in PLMs, where users provide rules for the system and retrieve outputs with degrees of truthfulness.

  .

- Chapter 5 proposes the idea of Type Embeddings, additional input embeddings that guide the model into the direction of a certain type, for example by steering the outputs to years instead of locations (Figure 1.1(D)). We present a method to compute TEs based on removing the first singular vector of a token embedding matrix. To test the effectiveness of this embedding, we offer a suite of tests. We study aspects related to type classification and sensitivity of a model equipped with the TE. We also perform a similar analysis in a layerwise fashion. Finally, we show that PLMs equipped with TEs provides an increase in performance on fact-retrieval datasets.

  > Chapter 5 focuses on gathering facts that match a user's intent, thus enhancing one aspect of the evidence retrieval stage in Figure 1.4. The novelty lies in encoding type information in PLMs without supervised learning, where users can provide typed tokens and steer the output type to match their needs. TEs can also be pre-computed and utilized by consumers.

- Chapter 6 analyzes the Birdwatch program, the first large-scale community-based fact-checking initiative by Twitter. We analyze how the crowd attempts to fact-check tweets in practice, while comparing with human experts and automated fact-checking tools. We focus on three aspects related to claim selection, evidence retrieval, and claim verification (Figure 1.3). Our insights show that the crowd could be effective in verifying truthfulness of claims in a faster pace compared to expert fact-checkers, however more care should be taken into profiling these workers, as in the uncontrolled environment we study (Twitter), since malicious users could manipulate the verification process in their favor. While a purely crowd-sourced approach to fact-checking has its limitations, it seems that most of these limitations can be complemented by a larger system incorporating experts and computational methods in the loop. We also release the matched dataset of 11.9k tweets with Birdwatch checks and identify 2.2k tweets verified both by Birdwatch users and experts.

  > Chapter 6 analyzes the Birdwatch fact-checking program through the lens of the pipeline in Figure 1.4. The contribution lies in providing empirical evidence that the performance of a crowd of non-experts matches, to some degree, that of experts in verifying tweets in an uncontrolled environment.

- Chapter 7 wraps up the results achieved in this thesis. We highlight some drawbacks of the proposed solutions and discuss several promising directions for future developments in the field.

1.4 Publications

This thesis is based on work done during the course of the PhD and published in peer-reviewed venues[6].

Chapter 2 includes work from the paper:

Gilbert Badaro, Mohammed Saeed, Paolo Papotti. **Transformers for Tabular Data Representation: A Survey of Models and Applications** (Survey Paper). Transactions of the Association for Computational Linguistics 2023.

Chapter 3 is based on the papers:

- Georgios Karagiannis*, Mohammed Saeed*, Paolo Papotti, Immanuel Trummer. **Scrutinizer: A Mixed-Initiative Approach to Large-Scale, Data-Driven Claim Verification**. (Research Paper) Proceedings of the VLDB Endowment Volume 13 Issue 12, August 2020, pages 2508–2521, Online.
- Georgios Karagiannis*, Mohammed Saeed*, Paolo Papotti, Immanuel Trummer. **Scrutinizer: Fact checking statistical claims**. (Demo Paper) Proceedings of the VLDB Endowment Volume 13 Issue 12, August 2020, pages 2965–2968, Online (Best Demo Paper Award at BDA 2020, 36ème Conférence sur la Gestion de Données).
- Mohammed Saeed, Paolo Papotti. **Fact-Checking Statistical Claims with Tables.** (Invited Paper). IEEE Data Engineering Bulletin Volume 44 Issue 3 2021.

Chapter 4 is based on the paper:

- Mohammed Saeed, Naser Ahmadi, Preslav Nakov, Paolo Papotti. **RuleBERT: Teaching Soft Rules to Pre-Trained Language Models.** (Long Paper) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing 2021, pages 1460-1476, Dominican Republic, Association for Computational Linguistics.

Chapter 5 is based on the paper:

- Mohammed Saeed, Paolo Papotti. **You Are My Type! Type Embeddings for Pre-trained Language Models.** (Long Paper) Findings of the Association for Computational Linguistics: EMNLP 2022, United Arab Emirates,

[6]Equal Contribution is denoted by a *.

Association for Computational Linguistics.

Chapter 6 is based on the paper:

- Mohammed Saeed, Maelle Nicolas, Nicolas Traub, Gianluca Demartini, Paolo Papotti. **Crowdsourced Fact-Checking at Twitter: How Does the Crowd Compare With Experts?** (Full Paper) 31st ACM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, Association for Computing Machinery.

I have also been involved in other publications during my thesis which are not part of this thesis:

1. I have supervised the application of SCRUTINIZER on The Shared Task on Evaluating Accuracy in Generated Texts challenge in INLG 2021 (Reiter and Thomson, 2020).

   Rayhane Rezgui, Mohammed Saeed, Paolo Papotti. **Automatic Verification of Data Summaries.** (Systems Paper) Proceedings of the 14th International Conference on Natural Language Generation 2021.

2. I have extended the baseline of FEVEROUS (Aly et al., 2021) with the addition of a neural re-ranker.

   Mohammed Saeed, Giulio Alfarano, Khai Nguyen, Duc Pham, Raphaël Troncy, Paolo Papotti. **Neural Re-rankers for Evidence Retrieval in the FEVEROUS Task** (Systems Paper) In Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER), pages 108–112, Dominican Republic. Association for Computational Linguistics

3. I was involved in work about generating ambiguous data for a model, where I implemented a baseline utilizing GPT-2.

   - Enzo Veltri, Donatello Santoro, Gilbert Badaro, Mohammed Saeed, Paolo Papotti. **Pythia: Unsupervised Generation of Ambiguous Textual Claims from Relational Data** (Research Paper) 39th IEEE International Conference on Data Engineering, California, USA .
   - Enzo Veltri, Donatello Santoro, Gilbert Badaro, Mohammed Saeed, Paolo Papotti. **Pythia: Unsupervised Generation of Ambiguous Textual Claims from Relational Data** (Demo Paper) Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA (Best Demo Paper Award).

2 Background

In this chapter, we provide the essential background knowledge upon which the succeeding chapters are based on. We start by reviewing Pre-trained Language Models (PLMs) in Section 2.1 by considering standard language models, attention, transformers, a knowledge-probing dataset for PLMs, and applications of transformer-based LMs on Tabular Data. We then consider Knowledge Graphs in Section 2.2. Finally, we explore fact-checking with human-in-the-loop in Section 2.3. More accompanying related work will be discussed in the following chapters.

2.1 Pre-trained Language Models

The *transformer* architecture (Vaswani et al., 2017), shown in Figure 2.1, has achieved great success in the natural language processing (NLP) domain. In this section, we present transformer-based language models, by first defining language models (Section 2.1.1), then we discuss transformers usage (Section 2.1.2), transformer components (Section 2.1.3), pre-training (Section 2.1.4), a prominent dataset to test factual and commonsense knowledge of PLMs (Section 2.1.5), and how tabular data is incorporated with PLMs (Section 2.1.6). In this section, we describe the vanilla transformer. A survey of notable transformer architectures can be found in (An, 2021).

2.1.1 Language Models

Definition. *Language Modeling* is the task of predicting what token comes next in a sequence of tokens. More formally, given a sequence of tokens $x^{(1)}, x^{(2)}, ..., x^{(t)}$, the goal is to compute the probability distribution of the next word , $x^{(t+1)}$, $P(x^{(t+1)}|x^{(1)}, x^{(2)}, ..., x^{(t)})$ where $x^{(t+1)}$ is restricted by a vocabulary of tokens V. A system performing this task is called a *Language Model*. Language Models (LM) are intended to distinguish grammatical from ungrammatical sequences in a specified language. In other words, given a phrase or a sentence in a language, a LM has to identify if it is fluent or plausible according to the grammar of that language or not. A language model is expected to identify "I defend

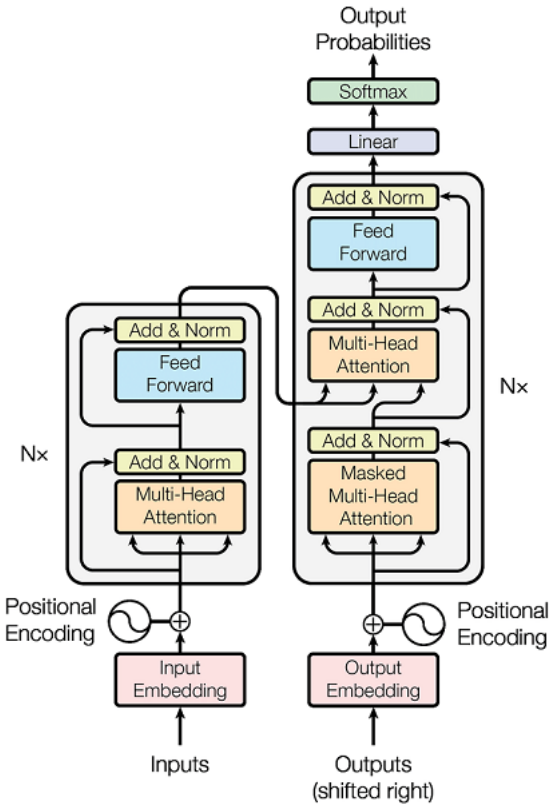

Figure 2.1: The Transformer Architecture (Vaswani et al., 2017)

my thesis." as a fluent sequence in English that accords with its grammar, whereas "my thesis defend I." as non-fluent or ungrammatical.

Brief History. Traditional approaches to language modeling included n-grams-based methods with the usage of smoothing techniques (Goldberg, 2017; Jurafsky and Martin, 2009). However, such approaches have drawbacks, mainly related to the costly scaling to longer n-grams. Nonlinear neural network models solved some drawbacks of traditional LMs, as they allow condition on larger context sizes with only a linear increase in the number of parameters. A popular model was proposed by (Bengio et al., 2000). Prominent architecture for LMs include RNNs (Elman, 2010), LSTMS (Hochreiter and Schmidhuber, 1997), GRUs (Chung et al., 2014), and most recently, transformer-based LMs (Vaswani et al., 2017). We stress the fact that all LMs discussed in this thesis are **transformer-based**.

2.1.2 Transformer Usage

The transformer architecture can be used as an encoder-decoder (Vaswani et al., 2017; Raffel et al., 2020), an encoder-only (Devlin et al., 2019; Liu et al., 2020b), or a decoder-only (Radford et al., 2019; Brown et al., 2020) model. The choice of the architecture

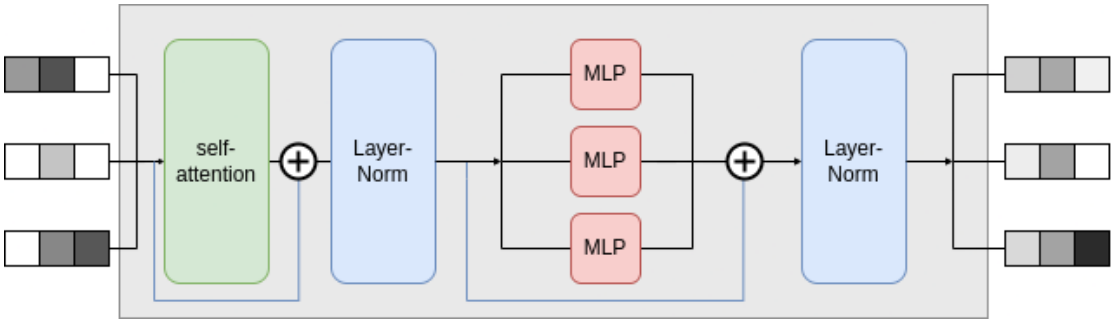

Figure 2.2: High-level overview of the components comprising a transformer encoder module.

depends on the final task. Encoder-only models are mainly used for classification. Encoder-decoder architectures are used for models that focus on sequence generation tasks such as machine translation. Decoder-only models are used for autoregressive tasks such as sequence generation.

2.1.3 Transformer Components

The vanilla *transformer* (Vaswani et al., 2017) is a sequence-to-sequence model (Sutskever et al., 2014) consisting of an encoder and a decoder, each of which is a stack of N identical modules. The encoder block is composed of a *multi-head self-attention* module (Section 2.1.3.1) and a *position-wise feed-forward network* (Section 2.1.3.2). The input is tokenized and embedded before inputting to the model (Section 2.1.3.3). *Residual connections* and *layer-normalization* modules (Section 2.1.3.4) are also used for stable training. Figure 2.2 shows a transformer encoder block.

2.1.3.1 Multi-Head Self-Attention

General Picture. The *self-attention* module is the fundamental operation of any transformer-based architecture. Simply, it takes as input a set of vectors, and outputs a contextualized version of that set of vectors, while conserving the same dimension. More formally, given a set of input vectors $x_1, x_2, ..., x_t$ of dimension d_{model}, the set of output vectors $y_1, y_2, ..., y_t$ are computed as $y_i = \sum_j w_{ij} x_j$. The weights w_{ij} are not parameters learned by the model, but rather derived from the input vectors x_i and x_j. They are often denoted as *attention weights*. A few more components, which will be discussed soon, are needed for a complete view of a transformer; but this is basically the fundamental operation of a transformer model. More importantly, this is the only operation in the whole architecture that propagates information between the vectors. Every other operation in the transformer is applied to each vector in the input sequence without interactions between vectors.

Computing Attention Weights. Each input vector x_i is represented in three different

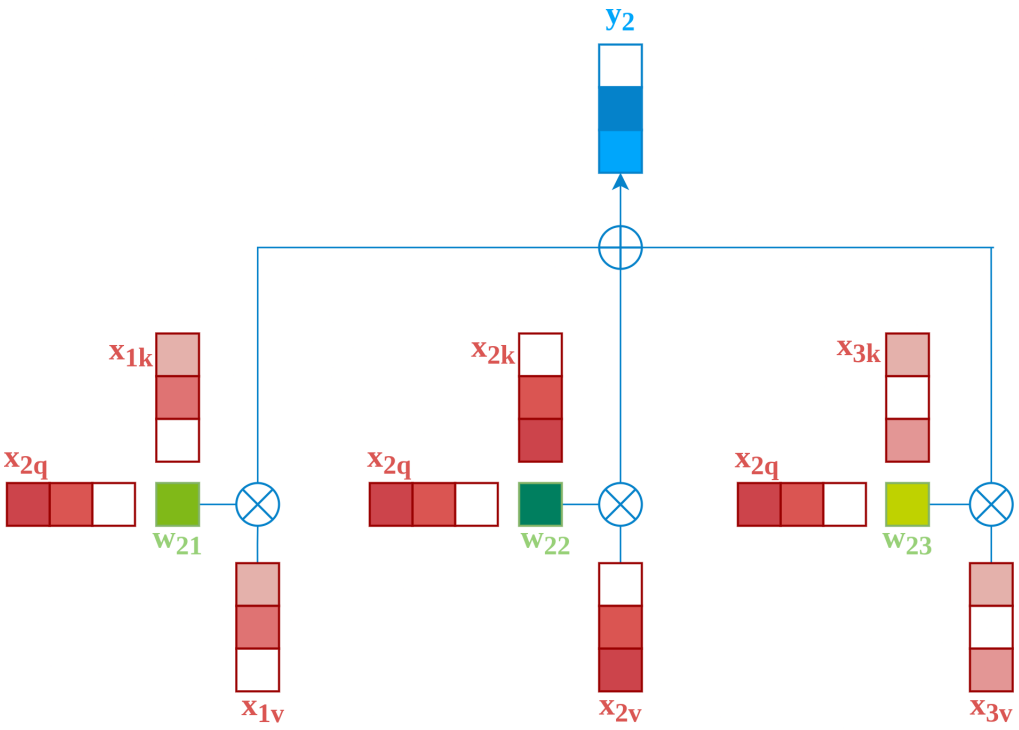

Figure 2.3: Diagram showing the self-attention mechanism. The value vectors (x_{1v}, x_{2v}, x_{3v}) are summed and weighted by the attention weights (w_{21}, w_{22}, w_{23}), which are computed from the key (x_{1k}, x_{2k}, x_{3k}), and query vector (x_{2q}) to produce a more contextualized representation (y_2). The self-attention mechanism is a permutation-invariant operation. For simplicity, the query, key, and value vectors have been drawn identically.

ways in a self-attention operation:

- It is compared to every other vector in the input to establish the attention weights for its own output y_i.
- It is compared to every other vector to establish the attention weights for the output of y_j.
- It is used as part of the weighted sum to compute each output vector once the attention weights have been established.

These roles are respectively called the *query*, the *key*, and the *value*. Computing such vectors is performed by a linear transformation with three matrices for the three respective representations (Eq. 2.1-2.3). $W_q \in \mathbb{R}^{d_{model} \times d_k}$ transforms an input x_i to its query representation q_i (Eq. 2.1), where d_k is the dimension of the key and query vectors. Similarly, $W_k \in \mathbb{R}^{d_{model} \times d_k}$ (Eq. 2.2) and $W_v \in \mathbb{R}^{d_{model} \times d_v}$ (Eq. 2.3) transform x_i to its key and value representations, respectively, where d_v is the dimension of the value vectors. This is followed by the dot product of the key and query vectors to compute the attention weights. To stop the dot product from growing too much, the dot product is scaled

(Eq. 2.4). A softmax operation is used to provide a normalized probability distribution over the attention weights (Eq. 2.5). Finally, a weighted sum of the value vectors yields the output of the attention mechanism (Eq. 2.6).

$$q_i = W_q x_i \tag{2.1}$$
$$k_i = W_k x_i \tag{2.2}$$
$$v_i = W_v x_i \tag{2.3}$$
$$w'_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}} \tag{2.4}$$
$$w_{ij} = softmax(w'_{ij}) \tag{2.5}$$
$$y_i = \sum_j w_{ij} v_j \tag{2.6}$$

A simplification of this mechanism is shown in Figure 2.3. This action performed by the self-attention mechanism resembles a "fuzzy" query operation on a key-value database. The matrix given by the attention weights w_{ij} is often called the *attention matrix*, as it portrays how much each element in a layer attends to each element in the previous layer. For inputs with large size, computing the attention matrix becomes infeasible and efficient methods are utilized (Tay et al., 2022).

Multi-Head Attention. The previous subsection described how a single attention mechanism or *attention head* works. Instead of relying on a single head, transformers use *multi-head attention*, where the original queries, keys, and values with dimension d_{model} are projected into d_k, d_k, and d_v dimensions, respectively, with h attention heads, each with a different set of learned projections. Each head output is computed with Eq. (2.6), and all outputs are concatenated and projected back to a d_{model}-dimensional representation with a learned transformation $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ (Eq. 2.7).

$$MultiHeadAttn(q_i, k_i, v_i) = Concat(head_1, \ldots, head_h)W^O \tag{2.7}$$

where $head_i$ is computed with Eq. (2.6).

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. It could be thought of an ensemble of classifiers where each head, while performing the same operation, contains different weight matrices, and could possibly perform different tasks. Indeed, certain heads could reflect certain syntactical aspects, such as heads where the head word attends to the dependent, or more complex semantic tasks such as coreference resolution (Clark et al., 2019b). However, some of these heads can be actually pruned without a significant reduction in performance (Michel et al., 2019).

Note that this kind of self-attention is found in the encoder module of a transformer. The same mechanism is found in the decoder module, with the exception that queries at a certain position can only attend up to that position inclusively. This is done by masking out all values in the input of the softmax which correspond to unwanted information

flow. Formally, the input in Eq. 2.5 is multiplied by a mask M that is set to $-\infty$ to prevent each position from attending to subsequent positions. This is usually called *masked self-attention*. For an encoder-decoder module, a special kind of attention is used where the queries are projected from the outputs of the previous (decoder) layer, whereas the keys and values are projected using the outputs of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This is called *cross-attention*.

2.1.3.2 Feed-forward network.

The position-wise feed-forward network (FFN) is a fully-connected feed-forward module operating separately on each position:

$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2 \tag{2.8}$$

where $ReLU$ refers to the rectified linear unit activation function (Fukushima, 1975). This FFN is applied to each vector from the output of the self-attention mechanism. Around two-thirds of the parameters in a transformer block are comprised in the FFN layer. Previous work posits that FFN layers act as key-value memories, where keys represent input patterns such as inputs ending with a specific word, or inputs encoding a "part of" relation, and values represent distributions over the input vocabulary V. FFN layers aggregate distributions weighted by scores computed from the keys (Geva et al., 2021b).

2.1.3.3 Input Embeddings

Before feeding the input to the self-attention layer, the input sentences are embedded. The vanilla transformer architecture embeds each token with a *token embedding* and a *positional embedding*.

Token Embeddings. Token embeddings are learned representations for each token in the vocabulary V. An input text is split into a sequence of tokens existing in vocabulary V. Tokenization could occur on a character-level, word-level, or subword-level. One of the most successful tokenization algorithms are the Byte Pair Encoding algorithm (Gage, 1994; Sennrich et al., 2016), used in GPT-3 (Brown et al., 2020), and WordPiece algorithm (Wu et al., 2016) in BERT (Devlin et al., 2019).

Positional Embeddings. Since transformers do not incur any form of recurrence, positional information is designated by absolute positional embeddings added to each token embedding of the transformer (Vaswani et al., 2017). Such embeddings can also be learned (Devlin et al., 2019). The purpose of the positional embedding is to allow a transformer to make sense of word ordering. Without it, the input would be perceived as a "bag of words". Other work proposes *relative* positional embeddings as a means of improving performance. Relative positional embeddings represent positional relationships between tokens instead of absolute positions of individual tokens (Shaw et al., 2018). A

combination of both absolute and relative positional embeddings has also been studied (Ke et al., 2021a).

Other Input Embeddings. Other input embeddings can be derived for specific purposes. For example, to incorporate tabular structure, row and column positional embeddings have been used to indicate the position of a cell in the table (Herzig et al., 2020). Other types include input embeddings for numerical input (Wang et al., 2021d).

2.1.3.4 LayerNorm and Residual Connections

Layer Normalization (LN) (Ba et al., 2016) and Residual Connections (He et al., 2016) are established methods to stabilize the training of deep neural networks. The LN is placed after the residual connections. (Xiong et al., 2020) has shown that placing the LN before the residual connections yields better training behavior.

2.1.4 Pre-Training

One of the key factors for the success of PLMs is *pre-training*. Pre-training finds better initial representations for the input as opposed to random initialization. This helps the model acquire a better starting point for embedding the input. In fact, with the increase in the number of parameters in a deep learning model, larger labeled datasets are needed for training. This poses a problem, as obtaining large-scale labeled data is cumbersome for several tasks due to expensive annotation costs. On the contrary, having access to large unlabeled data is relatively easy, and is leveraged through pre-training. Once pre-training is done, the model can be fine-tuned on a smaller (labeled) dataset targeted for a specific task.

A prominent example of a pre-training task would be the masked language modeling task (Devlin et al., 2019) where some random tokens are masked and are to be predicted by the model based on the context. A more detailed analysis of pre-training can be found in (Qiu et al., 2020). Examples of other pre-training tasks targeted for input tabular data can be found in (Badaro et al., 2023).

2.1.5 LAMA

LAMA (LAnguage Model Analysis) is a probe to assess how much knowledge is recalled by PLMs, and how different types of knowledge (such as facts about entities and common sense) affect their performance (Petroni et al., 2019). Simply, a PLM is asked to predict a masked word in a sentence, e.g., given the cloze sentence "*The theory of relativity was developed by [MASK].*", a PLM would predict the correct response *Einstein*. The cloze sentences used include relations between entities stored in Wikidata (from Google-RE corpus[1] and the T-Rex dataset (Elsahar et al., 2018)), common sense relations between concepts from ConceptNet (Speer and Havasi, 2012), and knowledge necessary for answering natural language questions in SQuAD (Rajpurkar et al., 2016).

[1] https://code.google.com/archive/p/relation-extraction-corpus/

2.1.6 Application on Tabular Data

To properly model the structure of data in tables, the vanilla transformers are extended and updated by modifying components at the (i) input, (ii) internal, (iii) output, and (iv) training-procedure levels. We briefly discuss the modifications here. A more detailed analysis can be found in our work (Badaro et al., 2023).

2.1.6.1 Input Level.

Modifications on the input level are usually designated with additional positional embeddings to explicitly model the table structure. Such embeddings improve performance on structure-related tasks. For example, embeddings that represent the position of the cell, indicated by its row and column IDs, are common for relational tables (Yin et al., 2020; Herzig et al., 2020). For tables without a relational structure, such as entity tables and spreadsheets, (Wang et al., 2021d) introduces tree-based positional embeddings to encode the position of a cell using top and left embeddings of a bi-dimensional coordinate tree.

2.1.6.2 Internal Level.

Most of the modifications on the internal level are applied to make the system more "structure-aware" by updating the attention module in a certain manner. For example, in (Yin et al., 2020) vertical self-attention layers are produced to capture cross-row dependencies on cell values by performing the attention module in a vertical fashion. Empirical results in (Wang et al., 2021d) show that employing row-wise and column-wise attention, instead of having additional positional embeddings for rows and columns, hurts model performance for cell-type classification tasks, but it is not the case for table-type classification tasks.

Other systems employ a masked self-attention module, which attends to structurally related elements such as those in the same row or the same column, thus ignoring the other elements, unlike the traditional transformer where each element attends to all other elements in the sequence (Deng et al., 2020).

Other modifications address the input size constraint of attention modules, where large tables are often neglected. Sparse attention methods are proposed to cope with this issue (Tay et al., 2022). For instance, (Eisenschlos et al., 2021) sparsifies the attention matrix to allow transformer heads to efficiently attend to either rows or columns.

2.1.6.3 Output Level.

Additional layers can be added on top of FFNs depending on the task at hand. Tasks such as cell selection (Herzig et al., 2020) and table meta-data prediction (Suhara et al., 2022) require training one or more additional layers. Classification layers for aggregation operations and cell-selection are used to support aggregating cell values (Eisenschlos et al., 2021; Yu et al., 2020). In (Liu et al., 2022b), aggregation operations are also "learned" end-to-end in a seq2seq task.

2.1.6.4 Training-Procedure Level.

Modifications on the training-procedure level can be attributed to the pre-training task and objective.

Pre-training Tasks Rather than applying traditional token-level masked-language modeling (MLM) on the serialized tabular data, effectively treating it as natural language sequences, most pre-training tasks are designed to consider the information about the table structure.

While some methods rely on the traditional masking of the tokens in the table cells, other tasks consider the masking of the whole cell independently of the number of tokens in it (Wang et al., 2021d; Yin et al., 2020). For instance, masking at the entity level enables the model to integrate the factual knowledge embedded in the table content and its context in its different forms. (Yin et al., 2020) also considers masking column names and data types for relational tables. In (Liu et al., 2022b), the pre-training task adopts an SQL engine to train the model to act as a neural SQL executor, thus enabling it to mimic the SQL semantics with relational tables.

Pre-training Objectives. The objective of the majority of the systems is to minimize cross-entropy loss for a certain classification task. (Wang et al., 2021b) utilizes a point-wise ranking objective for end-to-end training after pre-training, where multiple tables are ranked according to a relevance score given a certain query.

2.2 Knowledge Graphs

Knowledge graphs have emerged as an established tool for structuring knowledge over the web. They have played vital roles in several applications such as data integration and artificial intelligence. In this section, we briefly visit knowledge graphs, where we provide a quick definition (Section 2.2.1) and discuss the RDF Data Model (Section 2.2.2).

2.2.1 Definition

A knowledge graph (KG) is a directed labeled graph where the labels have well-defined meaning. The graph consists of labeled nodes and edges. Any entity or value can act as a node, where the edge connecting a pair of nodes captures the relationship between them. For example, a node representing `Paris` would be connected with the node representing `France` by the edge `CapitalOf` (Figure 2.4).

KGs are used in a variety of ways depending on the needs of an application, mainly for knowledge representation of real-world facts. The structured data not only enables "classic" analytics, such as querying with structured languages (such as SQL, SPARQL, or any graph oriented declarative language), but also advanced analysis, such as logical reasoning (Ahmadi et al., 2019; Ji et al., 2022). As KGs are modeled as graphs, they can be utilized for a plethora of tasks such as clustering (Saeedi et al., 2018), link prediction (Lin et al., 2015), topological sort (Pearce and Kelly, 2007), learning and using

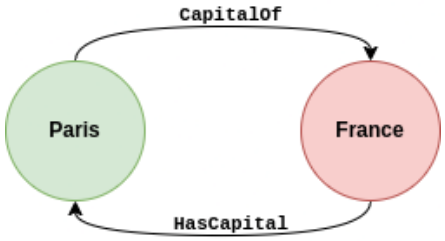

Figure 2.4: Two Nodes and their relations. `CapitalOf` is the inverse relation of `HasCapital`

embeddings (Wang et al., 2017), and rule-mining (Ortona et al., 2018b; Ahmadi et al., 2020). Examples of famous KGs include DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007), and Wikidata (Vrandečić and Krötzsch, 2014).

2.2.2 RDF Data Model

RDF. The Resource Description Framework[2] (RDF) is a framework for representing information on the web. The RDF data model and its query language SPARQL are standardized by the World Wide Web Consortium. A triple $p(s, o)$ following the RDF model, called an RDF triple, consists of a subject s, a predicate p, and an object o. A set of such triples is an RDF graph. For example, the fact that *Dante* was born in 1265 is expressed with a triple $birthYear(Dante, 1265)$. Nodes can exist in three flavors: IRIs, literals, and blank nodes. An IRI is an Internationalized Resource Identifier used to uniquely identify resources on the web. A literal is a value of a primitive data types such as number, date, and string. A blank node is a node representing a resource for which an IRI or literal is not given. It simply indicates the existence of a thing.

SPARQL. Is a recursive acronym for SPARQL[3] Protocol and RDF Query Language. It is a query language to retrieve and manipulate data stored using RDF, similar to SQL in relational databases. SPARQL contains capabilities for querying required and optional graph patterns, including conjunctions and disjunctions. The results of SPARQL queries can be sets or RDF graphs. For example, the following SPARQL query retrieves names of people born in Rome and died in Paris from DBpedia (Prefix definition has been omitted for brevity.):

```
SELECT ?name WHERE {
?person dbo:birthPlace dbr:Rome . ?person dbp:deathPlace dbr:Paris.
?person dbp:name ?name.
}
```

[2] https://www.w3.org/RDF/
[3] https://www.w3.org/TR/rdf-sparql-query/

2.3 Fact-Checking with Humans

As manual fact-checking cannot scale to the amount of generated fake news, automating certain aspects of fact-checking pipeline has taken a particular interest (Nakov et al., 2021a; Guo et al., 2022b). We discuss a few efforts in Section 2.3.1 where humans are mere consumers of the fact-checking system (Figure 1.2), and then follow by some work that includes humans in the fact-checking process as workers in Section 2.3.2.

2.3.1 What Technology has Offered Fact-Checking

Technology has helped automate some aspects of manual fact-checking for human acting as consumers of the fact-checking system:

Claim Check-worthiness. With the overflow of claims, deciding what claim is worthy fact-checking is inevitable. This has motivated the development of machine learning solutions (Elsayed et al., 2019; Nakov et al., 2021b). (Hassan et al., 2017b) first model this problem as a classification and scoring model, where the higher the score, the more likely an input sentence contains a check-worthy claim. (Konstantinovskiy et al., 2018) develop an annotation schema and a benchmark for automated claim detection. (Wright and Augenstein, 2020) model claim check-worthiness detection as a Positive Unlabeled learning instance (Elkan and Noto, 2008). Automating claim check-worthiness would reduce manual labor performed by fact-checkers.

Detecting Previously Fact-Checked Claims. As a fact-checked claim repeats itself, often in a paraphrased form and in other languages (Nakov et al., 2021a), it would be ideal to identify such claims beforehand. (Shaar et al., 2020) formulate the task by releasing two datasets and a proposing a learning-to-rank approach utilizing BERT and BM25 (Robertson et al., 1994). (Vo and Lee, 2020) debunk a tweet containing textual and visual features by retrieving relevant documents. (Ahmadi et al., 2022a) propose an unsupervised graph-based method for text matching, which is also applied to structural data. Detecting previously fact-checked claims would allow fact-checkers to focus on claims that have not been fact-checked.

Evidence Retrieval. As fact-checkers are flooded with external corpora for assessing a claim, narrowing down the list to fewer evidences to choose from is needed. The main approach for this utilizes BM25, or any other similarity score over vectorial representations between the input claim and the external corpora. The scores are then ranked in order to find the most relevant evidence. External corpora could be in the form of documents (Chen et al., 2017), relational tables (Aly et al., 2021; Saeed et al., 2021b), or even PLMs (Lee et al., 2020). Retrieval could also be on a passage level (Elsayed et al., 2019), or sentence level (Thorne et al., 2018b). Machine-based methods for evidence retrieval reduces time required to search for evidence by fact-checkers, especially with large unstructured corpora.

Automated Verification. Given pieces of evidence and a claim, a module for verdict prediction provides the final verdict of the input claim. This could be modeled as a binary

classification problem. (Thorne et al., 2018b) add a third label indicating that not enough information is included to produce a verdict. Other work tries to encode multi-class labels to account for the various shades of truthfulness utilized by fact-checkers (Wang, 2017; Augenstein et al., 2019). Usually, some model trained on a natural language inference dataset is used (Sundriyal et al., 2022; Aly et al., 2021). (Eisenschlos et al., 2020) perform "Table Entailment" which is about finding if a sentence is supported or refuted by the content of a table. (Clark et al., 2020) postulate that transformers can act as limited "soft theorem provers" over textual input. Particularly on social networks, one could benefit also from information of the network structure. (Bansal et al., 2021) utilize graph embeddings of follower-follower graphs on Twitter, alongside other features, to detect fake COVID-19 tweets. In addition to graph-based embeddings, (Gangireddy et al., 2020) use graph-based methods, such as biclique identification, for fake news detection on Twitter. (Shu et al., 2018) apply network diffusion models to trace the paths of fake news.

Justification Production. Justifying the verdict of a claim is a vital component, as expert fact-checkers need to convince others of their verdict. Not only that, but as humans, whether interacting as external users or as part of the system , obtaining explanations is key for a better human-machine interaction. Regarding automated fact-checking systems, justification production goes into the lines of interpretability and explainability of the used models. While some fact-checking systems, rely on logical rules that offer some form of explainability, most state-of-the-art methods utilize PLMs which are not explainable by design, as in other rule-based fact-checking systems (Gad-Elrab et al., 2019; Ahmadi et al., 2019). (Atanasova et al., 2020b) jointly train a PLM for predicting the veracity label and the explanation. (Tafjord et al., 2021) use an encoder-decoder T5 model (Raffel et al., 2020) to generate an answer to a hypothesis with the accompanying proof. (Atanasova et al., 2020a) analyze explainability techniques on downstream text classification tasks for neural models, and compare with human annotations. (Kotonya and Toni, 2020) survey the explainability of different automated fact-checking systems.

2.3.2 How can Humans be included as Workers?

Despite all this, fact-checkers agree that including humans in the fact-checking procedure as workers is inevitable (Arnold, 2020), as machine-based solutions have several limitations in the 'understanding' of language and in reasoning. Given these challenges, there is a clear opportunity of having humans involved along with machines in the process of fact-checking a claim, where humans could complement the deficiencies of machine learning models.

One approach to integrate humans in a fact-checking systems is through *Active Learning* (Vlachos, 2008; Konyushkova et al., 2017). The goal of active learning is to select which data points should be annotated in order to learn the model as quickly as possible. In practice, this means that instead of a costly annotation of the data, we select iteratively and adaptively which datapoints should be annotated next. (Karagiannis et al., 2020) is an example of such hybrid approaches for fact-checking utilizing the knowledge of experts to label certain data points chosen through an active learning algorithm to

minimize verification cost. The training procedure is steered through the labels given by the experts, which form a crucial part of the system, especially in domain-specific settings, where labeled data is scarce.

Another fundamental approach for integrating humans in fact-checking systems is *Crowd-sourcing*. The aim of crowd-sourcing is to gather a group of humans (crowd), who are not necessarily experts, to accomplish a certain task, under the premise that the wisdom of the crowd surpasses that of an individual. Amazon Mechanical Turk[4](MTurk) is the *de facto* standard crowd-sourcing website. Several works study and analyze the performance of the crowd while verifying claims. (Tschiatschek et al., 2017) utilizes labels by non-experts in a crowd-sourcing setting and applies Bayesian inference for detecting fake news while learning about a user's accuracy. (Qu et al., 2022) study how a PLM compares to a crowd of non-experts on truthfulness classification of claims, looking at both accuracy and confidence signals. (Fan et al., 2020a) propose fact-checking briefs, which are snippets of relevant text that accompany crowd-workers and fact-checkers to increase fact-checking quality. (Roitero et al., 2021) uses crowd-workers to verify coronavirus-related claims, and asks humans to provide evidence for their verdicts. (Nguyen et al., 2018) follows a mixed-initiative approach between a single user and a machine learning system where the machine finds relevant articles, infers the relevance of each, and predicts the final verdict, and the user can change the source reputation and stance of each retrieved article to correct the model. The authors find that human-machine interaction can be effective, but might end up misleading the crowd in cases where the machine-learning models performed poorly. Note that such crowd-sourcing experiments have been performed in controlled settings (such as MTurk) and on well-curated datasets (e.g. PolitiFact dataset (Wang, 2017)) which hide the intricacies of uncontrolled settings where interactions between the crowds are not as limited, and the crowd gets to choose which claims to verify from an increasing pool of unfiltered claims.

(Hassan et al., 2019) analyze an online fact-checking community on Reddit to find and verify check-worthy facts relating to US politics. Twitter, recently, proposed the Birdwatch program[5], a fully crowd-sourced approach for verification of claims in tweets. A Birdwatch user can create a note for a tweet, where a note comprises if the user's verdict, evidence, some metadata about the annotations. Other users can up/down vote such notes. Currently, the Birdwatch program is deployed in the US only. (Allen et al., 2021b) studies partisanship and behavior of users on Birdwatch. (Pröllochs, 2021) provide a holistic analysis of Birdwatch by studying aspects such as user profiles, reasons users report tweets, and agreement between the users. However, such work does not compare how such users perform when compared to expert fact-checkers regarding claim selection, evidence retrieval, and claim verification. We tackle these in Chapter 6.

[4] https://www.mturk.com/

[5] https://blog.twitter.com/en_us/topics/product/2021/introducing-birdwatch-a-community-based-approach-to-misinformation

3 Fact-Checking Domain-Specific Statistical Claims with Relational Tables

Organizations spend significant amounts of time and money to manually fact-check text documents summarizing data. In Figure 1.1(C), we have seen that some claims require aggregating content from a database. We also stated the difficulty when multiple claims come into play, where multiple human interventions are needed, but is not clear what is the optimal procedure. In this chapter, we introduce Scrutinizer, a system to reduce verification overheads by supporting human fact-checkers in translating text claims into SQL queries on a database. Scrutinizer coordinates teams of human fact-checkers. It reduces verification time by proposing queries or query fragments to the users. Those proposals are based on claim-text classifiers, trained on real world data (i.e. not synthetic), that gradually improve during the verification of a large document. In addition, Scrutinizer uses tentative execution of query candidates to narrow down the set of alternatives. The verification process is controlled by a cost-based optimizer. It optimizes the interaction with users and prioritizes claim verification. For the latter, it considers expected verification overheads as well as the expected claim utility as training samples for the classifiers. We evaluate the Scrutinizer system using simulations and a user study with professional fact-checkers, based on actual claims and data. Our experiments consistently demonstrate significant savings in verification time, without reducing result accuracy.

3.1 Introduction

Data is often disseminated in the form of text reports, summarizing the most important statistics. For authors of such documents, it is time-consuming and tedious to ensure the correctness of each single claim. Nevertheless, erroneous claims about data are not acceptable in many scenarios, as each mistake can have dire consequences. Those consequences reach from retractions (in case of scientific papers (Hosseini et al., 2018)) to legal implications (in case of business or health reports (Ash et al., 2004)). Besides the authors, erroneous claims can have serious consequences for the target audience (Brainard and Hunter, 2020). We note that approaches that integrate table structure into PLMs,

Index	2017	2018	...	2030	2040
PGElecDemand	22 209	22 793	...	29 349	35 526
PGINCoal	2 390	2 412	..	2 341	2 353
TFCelec	21 465	22 040	...	28 566	34 790
...	...	...	...	...	...

Figure 3.1: Global Energy Demand history and estimates (GED); the full table has 22 rows and 70 attributes.

similar to those in Section 2.1.6, would not be a good fit to the problem, as the complex queries to fact-check a claim would not be resolved solely by such approaches. Also, such approaches consume the entire table as an input, which is infeasible for industrial settings where having tables with large number of rows and columns is standard. In addition, PLMs performing Question Answering on Tables (Herzig et al., 2020), Semantic Parsing (Yin et al., 2020), or Table-based Fact-checking (Liu et al., 2022b) either do not consider aggregation of cell values, or have been trained on a pre-defined set of functions and require training from scratch to accommodate new aggregation functions. As obtaining labeled data is cumbersome, this necessitates the need for a human-in-the-loop component. Besides, in real-world scenarios, training data would not be available, and having humans label the data is a crucial part of the fact–checking system that is not found in such approaches. For this, we present Scrutinizer, a "mixed-initiative" fact-checking system that helps teams of fact-checkers to verify consistency of text and data efficiently.

Our work is inspired and motivated by two real-world use cases. We describe those use cases in the following and use them as examples and benchmarks throughout the chapter.

Use Case 1. *The International Energy Agency (IEA) is a Paris-based intergovernmental organization. Every year the agency produces a report of more than 600 pages about the energy consumption and production in the world, covering historical facts and predictions both for individual countries and at the world level. We have been given access to the 2018 edition, which contains 7901 sentences with 1539 manually checked statistical claims. IEA requires each claim to be verified independently by three persons (beyond the claim text author). Hence, verification takes months of work of a team of domain experts.* Scrutinizer *is the first result of a collaboration aimed at reducing time and financial overheads of that verification process.*

Consider the following example claim from that scenario.

Example 1. *The IEA database contains hundreds of relational tables with information about energy, pollution, and climate. A fragment of a table is reported in Figure 3.1. Consider the claim "**In 2017, global electricity demand grew by 3%**, more than any other fuel besides solar thermal, reaching 22 200 TWh.". An expert validates the*

claim in bold by identifying the relevant table(s) and by writing a query over such table to collect the relevant information. In the example (assuming unique values in the `Index` *column):*

```
SELECT POWER(
 (SELECT 2017 FROM GED WHERE Index='PGElecDemand')/
 (SELECT 2016 FROM GED WHERE Index='PGElecDemand'),
 1/(2017-2016)) -1
```

Finally, the expert compares the output of the query with the claim and either validates or updates the claim.

Unlike the first, our second use case assumes verification by non-expert, anonymous crowd workers.

Use Case 2. *The spread of the Coronavirus is accompanied by a spread of misinformation. This "Infodemic" (Latvian Public Broadcasting, 2020) is testing manual verification capacities of large social media platforms (Horwitz, 2020), thereby motivating automated methods. Some (even though not all) misclaims about the novel Coronavirus refer to numerical statistics, thereby falling within the scope of the* Scrutinizer *system. For instance, misclaims about absolute or relative case counts for specific regions and time ranges are common (Wikipedia, 2020). We therefore decided to create a public Web interface (Karagiannis et al., 2020), based on the* Scrutinizer *system, that allows to verify statistical claims on the Coronavirus. The database used for verification consists of daily updated statistics from official sources such as the World Health Organization (WHO) and the Center of Disease Control (CDC). Our Website, recently covered in the press (Claveau, 2020; Milucci, 2020; Spadaro, 2020), has attracted over 12,000 distinct users at the time of writing.*

In both scenarios, we verify claims given as natural language text. We verify or refute a claim by formulating a query on an associated database. Of course, the primary challenge is to translate natural language text into SQL statements. This problem has been the focus of significant prior work (Agrawal et al., 2002; Li and Jagadish, 2016; Iyer et al., 2017; Li and Jagadish, 2014; Zhong et al., 2017; Saha et al., 2016; Weir et al., 2019). The scenarios we consider are however specific and require a novel approach. First, prior work typically aims at automating text to query translations. Relying on purely automated translation would however not be acceptable for IEA for instance, due to the high stakes and limitations of current technology. Hence, we verify claims in a mixed-initiative approach that integrates human feedback. Second, prior work aims at translating queries instead of claims. This matters, as we can rank queries based on how close their results come to claimed values. Finally, prior work typically considers single queries. For example, in the case of IEA, we rather deal with large documents that contain hundreds of related claims. This can be exploited to reorder claims for verification and minimize expected verification overheads. Literature on data-driven claim verification (Cao et al., 2018; Ibrahim et al., 2019; Jo et al., 2019), discussed in more

detail in Section 3.7, is more sparse at this point. In our experiments, we demonstrate that prior work cannot address the use cases we consider.

When mapping claims to queries, SCRUTINIZER considers a scenario-specific query search space. This search space is defined by a domain expert with SQL training and knowledge of the database schema (e.g., the head of the fact-checker team in case of IEA). The same person supplies text snippets that allow SCRUTINIZER to formulate natural language questions about query properties. The bulk of the verification work is done by another group of fact-checkers. Those can be domain experts without IT background (e.g., the fact-checker team at IEA) or a crowd of anonymous users (like in our second use case). To map claims to queries, SCRUTINIZER introduces a scenario-specific set of classifiers. Each classifier is associated with a choice point in the query search space. This can be the choice between multiple query templates (e.g., selecting the arithmetic formula in Example 1) or filling in a placeholder (e.g., the year 2016 in Example 1). Having the result of each classification for a given claim can determine the associated query. However, if classifier confidence is below a threshold, classification results must be verified by human workers.

Our goal is to minimize overheads for human workers. Those overheads are determined by multiple factors. We expect that verifying a correct option for a query property (e.g., a proposition for an arithmetic formula) is typically faster, for workers, compared to suggesting the correct answer from scratch. We verify this intuition by a user study in Section 3.6. Hence, verifying claims based on high-quality classifier suggestions is cheaper than for low-confidence claims. On the other side, obtaining the correct answer for low-confidence claims and using it as training data, may improve classifier accuracy for the remaining claims. We take both factors into account when determining the order in which claims are verified. Also, we optimize the sequence of questions asked to workers about specific claims. Here, we can exploit the fact that getting answers for certain query properties implicitly prunes options for others.

SCRUTINIZER generates suggestions for query translations via classifiers. In principle, various types of classifiers can be used. Currently, we use PLMs (in particular XLM model (Conneau and Lample, 2019a)) with task-specific fine-tuning. Besides the claim text, we also exploit query evaluation results to rank query candidates. More specifically, to break draws between likely query options, we rank queries higher if their numerical evaluation results match numbers that appear in claim text. In our experiments, we demonstrate that SCRUTINIZER outperforms various baselines for all considered scenarios.

In summary, the original scientific contributions of this chapter are the following:

- We introduce the problem of mixed-initiative data-driven fact-checking and two corresponding real-world use cases (Section 3.2).
- We describe the SCRUTINIZER system (Section 3.3), featuring semi-automated claim to query translations (Section 3.4) and planning modules for optimally interacting with fact-checkers (Section 3.5).
- We demonstrate experimentally that SCRUTINIZER performs well in several complementary scenarios, improving over purely manual fact-checking as well as automated baselines (Section 3.6).

3.2 Problem Model

SCRUTINIZER aims at mixed-initiative verification (MIV) of statistical claims from relational data. The term "mixed-initiative" refers to the fact that our approach combines feedback from human workers with automated verification.

Definition 2. *One instance of the **MIV Problem** is defined by a quadruple $\langle \mathcal{T}, \mathcal{P}, \mathcal{Q}, \mathcal{L} \rangle$. Here, $\mathcal{T}$ is the verification target. It describes the claims to verify, and the database used for verification. Parameters $\mathcal{P}$ are configuration parameters. They constrain verification and contain constants used for cost-based planning. $\mathcal{Q}$ is the query template. It describes complete SQL queries to consider during verification. Finally, $\mathcal{L}$ maps template components to text snippets, used for formulating associated questions to crowd workers. A solution to MIV maps each input claim to a Boolean verification result and to an SQL query justifying that result.*

The goal is to reduce overheads for human fact-checkers in MIV. Optionally, the input also contains training data mapping prior claims from the verification domain to corresponding queries. This is however not required ("cold start scenario") as training data is dynamically collected during verification. Queries can validate claims in different ways. Claims may explicitly mention numbers that can be matched against query results. We call this claim category "Explicit Claims" (some prior work is specific to that category (Jo et al., 2019)). Alternatively, claims may be verified by queries returning a Boolean value, with the semantics that the claim holds if that value is true ("Implicit Claims"). Next, we define the four components from Definition 2.

Definition 3. *A **Verification Target** $\mathcal{T} = \langle C, D, s \rangle$ is a set C of claims to verify, a relational database D used for verification, and (optionally) a function $s : C \mapsto \mathcal{S}$ mapping claims to document sections. Each claim $c \in C$ is defined by a natural language description, containing the claim and relevant context. Each claim can be verified or refuted by running an associated query on database D.*

Definition 4. ***Configuration Parameters** $\mathcal{P}$ include constants representing cost estimates for different types of actions performed by human checkers. The specific parameters and their semantics will be described in Section 3.5. Also, they include a confidence threshold ρ for verification. Human workers are only used for claims where the confidence of automated verification is below ρ. Finally, a parameter determines the number of workers asked the same question.*

Definition 5. *A **Query Template** describes a space of SQL queries to consider during verification. We distinguish **Complete Templates**, describing complete SQL queries, from **Fragment Templates**, describing query parts. We define templates recursively. Any SQL query fragment, using schema elements from the target database, or an SQL keyword alone, is a (constant) template. Assuming that $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ are templates, then $CAT(\mathcal{Q}_1, \ldots, \mathcal{Q}_n)$ is also a template. It represents the* concatenation *of queries (or query fragments) matching templates $\mathcal{Q}_1$ to $\mathcal{Q}_n$. Similarly, $CHC(\mathcal{Q}_1, \ldots, \mathcal{Q}_n)$ represents a*

choice *among different templates. A query matches the choice template if it matches at least one of the templates $\mathcal{Q}_1$ to $\mathcal{Q}_n$. Finally, if $\mathcal{Q}_1$ is a complete query template describing queries that evaluate to scalar numerical results then $EXP(\mathcal{Q}_1)$ is a template as well. It matches all free arithmetic* expressions *that can be formed using queries matching $\mathcal{Q}_1$ (and constants) as operands, and logical and arithmetic operators (but no other SQL keywords such as* `FROM` *or* `SELECT`*). Templates using concatenation (CAT), choice (CHC), or free expressions (EXP) are* ***Composite Templates****. We call templates used as operands for concatenation, choice, or free expressions (here: $\mathcal{Q}_1$ to $\mathcal{Q}_n$) their* ***Components****.*

Example 2. *We instantiated* Scrutinizer *in a public Web interface ("CoronaCheck") for verifying single statistical claims about the novel Coronavirus, submitted by users, using data from WHO and CDC. Data is represented as tables, containing numbers (e.g., the number of confirmed cases) for different time periods in different columns, and data for different regions in different rows. For instance, we support comparative claims about the number of confirmed cases. The corresponding query space is described by a template of the form $CAT('SELECT', SQ, '>', SQ)$ where SQ is a template matching lookup queries, retrieving the number of cases for specific regions and times. SQ is defined as $CAT('SELECT', T, 'FROM\ NrCases\ WHERE', CP)$, where $T = CHC('Jan', 'Feb', \ldots)$ is a choice over time periods (represented as column names) and $CP = CAT(CN, '= Cname')$ models restrictions of the region (i.e., an equality predicate on the country name column). E.g., the query*

```
Select
 Select Feb From NrCases Where Cname='Germany' >
 Select Jan From NrCases Where Cname='Germany'
```

matches that template. Overall, we use a choice template for verification. Each component represents queries used to verify one popular type of claim, including the one described above.

Our goal is to map claims to queries. If automated verification fails, we solicit feedback from human fact-checkers. The labeling function allows us to formulate questions to workers for narrowing down queries for a given claim.

Definition 6. *The* ***Labeling Function*** *$\mathcal{L}$ maps query template components to text for formulating questions. It maps each choice element (CHC) to a question text, and, optionally, each choice option to a human-readable label. Also, it maps each formula element (EXP) to question text, asking workers to select an appropriate formula. No labels are needed for constant query fragments and concatenations. We also call choice and expression elements* choice points *or* query properties*, as they entail claim-specific decisions.*

Next, we show how to instantiate this model for our two use cases: verifying IEA and Coronavirus claims.

Example 3. *At IEA, claims are extracted from reports by the head of the fact-checker team (potentially adapting claim text slightly to make claim context more apparent).*

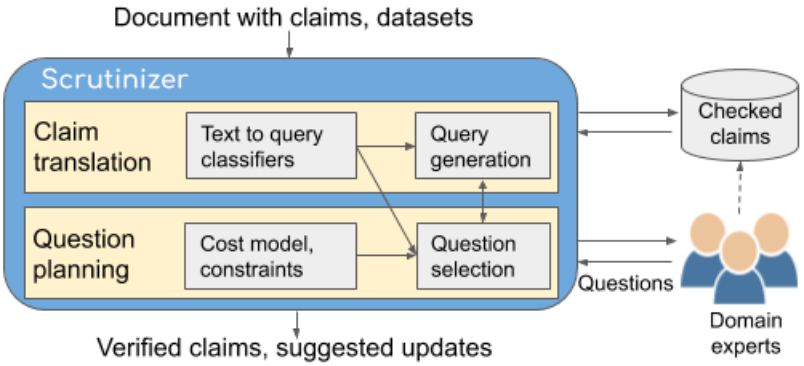

Figure 3.2: Architecture of SCRUTINIZER.

Claims are verified from a database containing hundreds of tables containing historical data (e.g., on energy consumption and production) as well as simulation results. Claims can be verified using the query template $CAT('SELECT', EXP(SQ))$. *Component template SQ models data lookups and is of the form* $CAT('SELECT', YR, 'FROM', TB, 'WHERE', CD)$. YR *denotes a specific year (generally modeled as columns by IEA),* TB *the table name (associated with a specific scenario such as global forecast of CO2 emissions), and* $CD = CAT('Index =', PV)$ *a condition fixing the primary key column (generally called "Index") to a specific value. We model formulas using the year itself as a numerical operand via a sub-query accessing a special identity table (mapping each year to itself). The labeling function assigns for instance* YR *to the question text "What is/are the corresponding year(s)?" that is shown to crowd workers to solicit feedback (see Figure 3.3). Also,* $\mathcal{L}(EXP(SQ)) =$*"Enter a formula translating the claim!". For highest accuracy, IEA requires manual verification of each claim (a confidence threshold of* $\rho = 100\%$*). The added benefit of* SCRUTINIZER *lies in suggesting likely verification options to save checking time. As a final step for each claim, workers are asked whether the associated query (with evaluation result) validates or refutes the claim.*

The query template above matches the query from Example 1. Here, the free expression uses the `POWER` function and four instances of SQ (two lookups and two year values, internally modeled as lookups in the identity table). CoronaCheck does not yet introduce new expressions on the fly, based on user feedback, while we consider this possibility for the IEA scenario. Hence, we use fixed formulas in the CoronaCheck templates but a free expression (EXP) element for the IEA template.

Example 4. *The query template for CoronaCheck was described in Example 2. If classifier confidence is above a threshold of* $\rho = 0.9$*, users are shown a Boolean verification result. Clicking on the result reveals query and data used for verification. If confidence is below the threshold, users are shown a dialog soliciting them to select query and data themselves (by selecting from given answer options). Based on crowd feedback,* SCRUTINIZER *improves accuracy over time.*

Algorithm 1: Main verification algorithm.

```
   // Verify claims C using models M and return verification results.
1  Function Verify(C, M):
       // Initialize verification result
2      A ← ∅
       // While unverified claims left
3      while C ≠ ∅ do
           // Select next claims to verify
4          N ←OptBatch(C, M)
           // Select optimal question sequence
5          S ←OptQuestions(N, M)
           // Get answers from fact-checkers
6          A ← A∪GetAnswers(N, M, S)
           // Retrain text classifiers
7          M ←Retrain(N, M, A)
           // Remove high-confidence claims
8          C ← C\Verified(N, M, A)
       // Return verification results
9      return A
```

3.3 System Overview

Figure 3.2 shows a simplified overview of Scrutinizer. The system encompasses two primary components. The automated translation component leverages machine learning to identify the elements that define every claim, i.e., candidates for datasets, attributes, rows, and comparison operations. The question planning component interacts with human domain experts to verify such elements and the checking results, optimizing verification tasks for maximal benefit.

Verification comprises the following high-level steps. First, based on the query template (Definition 5), we create a set of text classifiers. We introduce classifiers for choice points in the query template, as detailed in Section 3.4.1. Those classifiers map claim text to query properties. If available, they are trained with pairs of claims and queries from prior verification sessions. Second, as an alternative to automatic classification, we generate a set of questions on query properties about each claim. Those questions can be asked, optionally, to human fact-checkers, if classification confidence is below a threshold. Section 3.4.2 provides details on question generation.

After those preparatory steps, an iterative verification algorithm starts. It runs until all claims have been verified with sufficiently high confidence. The corresponding pseudocode in Algorithm 1 is simplified for readability and contains only the most important parameters (e.g., we do not explicitly refer to configuration parameters $\mathcal{P}$).

In the simplest case, based on previous training data, we are able to map each claim

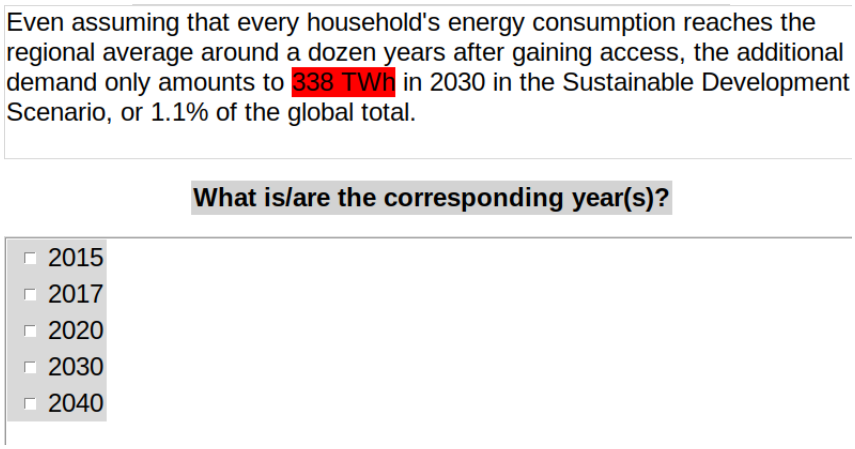

Figure 3.3: Screen soliciting workers to select relevant years for marked up claim.

to a query with sufficiently high confidence. Typically, this is not possible. For instance, IEA generally requires utmost precision in their reports. To achieve an accuracy close to 100%, inspection by human workers is generally required. Scrutinizer therefore interacts with human workers, asking them to verify classifier results or to suggest alternatives. Figure 3.3 shows an example screen (the lower half of the screen, asking workers to enter new years if needed, was cropped). We submit claims to workers in batches. Optionally, the same claims are verified by multiple workers (the number of workers can be configured). If so, we take the majority vote in case of conflicting answers. For free expressions, we use a simple normalization method (e.g., we remove spaces and capitalize symbols) before comparing answers of different workers. Depending on the scenario, we may be able to choose the order in which claims are verified. For IEA for instance, we can choose which out of hundreds of claims in a given report are verified first. We prioritize claims in a principled manner, outlined in Section 3.5.2, to minimize expected verification cost. We take into account the expected verification cost (which is lower if classifier suggestions are expected to be accurate) as well as utility in improving classifier precision. The latter point applies since answers from human workers can be used immediately as additional training data, thereby benefiting the verification of remaining claims.

Validating claims is easier for humans if the system suggests correct answer options for questions on query properties. As described in detail in Section 3.4.3, we rank likely queries (according to our classifiers) based on their query evaluation results. If presenting workers with answer options, we show them in order of likelihood to minimize expected overheads. Section 3.5.1 contains more details on how we select and prioritize questions asked about specific claims.

Algorithm 2: Initializing ML classifiers.

```
   // Get classifiers for choice points in query template 𝒬.
1  Function InitML(𝒬):
       // Distinguish type of query template
2      if 𝒬 is constant then
3          return ∅ // No classifiers needed
4      else if 𝒬 = CAT(𝒬_1, ..., 𝒬_n) then
           // Combine classifiers of components
5          return ∪_{1≤i≤n}{InitML (𝒬_i)}
6      else if 𝒬 = CHC(𝒬_1, ..., 𝒬_n) then
           // Classifiers for choice and components
7          return ∪_{1≤i≤n}{InitML (𝒬_i)} ∪ {M_CHC(𝒬)}
8      else if 𝒬 = EXP(𝒬_S) then
           // Classifiers for expression and component
9          return {InitML(𝒬_S)} ∪ {M_EXP(𝒬)}
```

3.4 Claim Translation

We exploit three sources of information to translate claims to queries. First, we introduce classifiers that use claim text as features. Second, we request feedback from human fact-checkers. Third, we use evaluation results from candidate queries to rank them by likelihood. The following subsections discuss those mechanisms in more detail.

3.4.1 Claim Classification

For a given scenario, the query template (see Section 3.2) defines the search space for queries. It defines degrees of freedom in the form of choice points (i.e., one out of several options is correct) and free expressions. We introduce one classifier for each of those query properties. Those classifiers exploit claim text as well as the claim location (e.g., the claim section for IEA reports) as features. Different types of classifiers can be used. We exploit pre-trained language models that are fine-tuned by task-specific training (Conneau and Lample, 2019a).

Algorithm 2 illustrates which classifiers are created for a given query template. This is done before the actual claim verification starts. Algorithm 2 takes the query template as input, the output is a set of classifiers linked to specific choices in the query template. Function InitML decomposes input templates and collects classifiers for components via recursive invocations. The behavior of the function depends on the type of input. No classifiers are necessary for constant query templates. For concatenations, we take the union of the classifiers for the concatenation operands (the components). Similarly, we union classifiers from components for choice elements together with a new classifier (initialized with function $\mathcal{M}_{CHC}(\mathcal{Q})$) in charge of selecting one of the choice options. For free expressions, we collect classifiers for the component and introduce a new classifier

Template Type	CAT	CHC	EXP
Answer Multiplicity	$[l, u]$	$[0, u]$	$[0, \infty]$

Table 3.1: New bounds on answer multiplicity given components with bounds $[l, u]$.

to determine the expression itself (function $\mathcal{M}_{EXP}(\mathcal{Q})$). After initialization, classifiers are trained during verification, using input from human fact-checkers. Optionally, they can be trained before verification starts if training data is available, e.g, in case of IEA, labels from prior text versions.

Example 5. *Consider the query template from Example 3 (IEA). The template is defined as a concatenation between a constant and a free expression. The free expression, in turn, depends on a component that represents an SQL query. Here, we introduce a classifier determining the free expression, a classifier determining the table TB within the sub-query, one for the year YR, and one for the row index PV.*

3.4.2 Generating Questions

Classifier accuracy is limited as automated natural language understanding is imperfect. Also, classes may be initially unknown, e.g., a claim whose associated query uses an expression that has not appeared before. To make up for the limitations of automated translation, we may request feedback from human workers. We introduce questions for each property of the query template. This means that each classifier is complemented by a question. Its answer, for a specific claim, replaces the classification result. Note that all those questions are optional. We may not ask about a query property if the confidence of the corresponding classifier is sufficiently high. Also, the answer to one question may prune possible answers for another question about the same claim. We discuss in Section 3.5 how to select which questions/answers to expose in which order. Here, we discuss how to generate the set of possible questions.

We omit the code for generating questions as it is similar to the one for generating classifiers (Algorithm 2) and describe the differences instead. For each initialized classifier, a corresponding question is introduced. The question uses text snippets, given via the labeling function $\mathcal{L}$ (see Section 3.2). For context, workers are shown the claim text along with the question. Furthermore, workers can select between multiple ranked answer options. In that context, we must decide how many answers workers can select. For instance, given a choice between constant query fragments, only one single answer can be accepted. On the other side, free expressions may contain a number of symbols that is *a-priori* unknown. Hence, unless the concrete expression is known, an arbitrary number of selections can be made for any answers that refer to the component template. We calculate lower (l) and upper bounds (u) on allowed answer multiplicity in a bottom-up approach with the recursive rules in Table 3.1. Given a question about a query template with answer multiplicity in $[l, u]$, it shows multiplicity after inserting the template as a component into a composite template.

Algorithm 3: Generating candidate queries using classifiers.

```
// Generate most likely queries matching Q, using k most likely
   alternatives per choice according to ML classifiers M, for claim
   c.
Function GQ(Q, M, k, c):
    // Distinguish type of query template
    if Q is constant then
        return Q
    else if Q = CAT(Q_1, ..., Q_n) then
        // Combine most likely queries from components
        // return GQ (Q_1, M, k, c) × ... ×GQ (Q_n, M, k, c)
    else if Q = CHC(Q_1, ..., Q_n) then
        // Most likely queries for most likely choices
        return {GQ (Q_S, M, k, c)|Q_S ∈ M(Q, k, c)}
    else if Q = EXP(Q_S) then
        // Get most likely operands
        O ←GQ(Q_S, M, k, c)
        // Most likely expressions using operands
        return {e.substitute(O)|e ∈ M(Q, k, c)}
```

Example 6. *Consider the sub-query element (SQ) from Example 3. It must contain exactly one year, hence we have multiplicity bounds* $[1, 1]$ *for questions on the year as long as we consider SQ alone. However, SQ appears inside a free expression. This changes the multiplicity range to* $[0, \infty]$ *according to Table 3.1. This means that, when asking workers which years are relevant for a claim (see Figure 3.3), we allow them to select arbitrarily many options.*

Beyond questions on specific query properties, manual verification of each claim ends with a question asking workers to validate the query as a whole (i.e., all properties) and to determine the final claim verification result.

3.4.3 Query Generation

Besides the claim text, we also use query evaluation results to rank queries. This ranking determines the order in which options are shown to workers (see Section 3.5). For Boolean queries, a true result typically indicates that the query verifies the claim. Hence, as a heuristic, we rank queries returning "True" above queries returning "False" (or "Null"). This assumes that accurate claims are more likely than inaccurate claims, an assumption that has been used in prior work (Jo et al., 2019) and holds for our test scenarios. Queries with numerical result typically validate a claim by calculating numbers that appear in claim text. Hence, we heuristically rank queries by the distance between query result and numbers that appear in claim text (we use the number with minimum distance if

the claim contains multiple numbers).

Of course, we cannot evaluate all possible queries matching the template. Instead, we focus on queries that are likely according to classifiers and worker answers (if any). Algorithm 3 shows how we derive the scope of queries for evaluation. The pseudocode is simplified, assuming that only classifier results but no worker answers are considered. Worker answers may prune classes further.

Algorithm 3 uses the scenario-specific query template, the trained classifiers, the current claim, and an integer parameter k as input. Parameter k determines how many of the most likely alternatives to consider for each query property. Function $M(\mathcal{Q}, k, c)$ retrieves the k most likely alternatives for the classifier associated with query template $\mathcal{Q}$, considering the text and location (section) of claim c. The algorithm composes likely queries for a template by combining likely queries for its components (obtained via recursive invocations). For instance, for concatenation, likely queries are formed via cross product between likely queries for components. For free expressions, likely queries are created in two steps. First, we retrieve a likely set of operands (Line 16) via recursive invocation. For the k most likely operands, we instantiate the k most likely expressions in all possible ways. I.e., we substitute their symbols with most likely operators, considering all possible permutations.

Example 7. *Consider the query template from Example 3 again. Assume we obtained formula* `POWER(a/b,1/(c-d))-1` *as free expression (either by classification with sufficiently high confidence or by asking human workers). Furthermore, assume that the most likely operands for the formulas are the years 2016, 2017, and 2018, and two sub-queries retrieving energy demand data for years 2016 and 2018, respectively. We evaluate queries* $5 \cdot 4 \cdot 3 \cdot 2 = 120$ *queries, considering all possible substitutions via likely operators. As the claim text (see Example 1) contains the number 3%, we rank by the distance between query result and that number. When proposing answer options to workers, we order them according to the rank of the query they appear in.*

3.5 Question Planning

Question planning consists of two tasks: determining optimal questions to verify single claims, and determining an optimal verification order between claims. We discuss the first problem in Section 3.5.1 and the second one in Section 3.5.2.

3.5.1 Single Claim Verification

For each claim, we generate a series of screens (Figure 3.3 shows an example). Each screen contains questions that are answered by a worker. Each screen is associated with one specific query property. On the upper part of each screen, workers are shown a set of answer options with regards to the current property. Those answer options are obtained from our classifiers. On the lower part of each screen, workers have the option to suggest new options, if the correct answer is not on display. The final screen for each claim asks

directly for the query translating the current claim. Answers to prior questions may have allowed us to narrow down the range of possible queries. If so, the chances for confronting workers with the correct query increase.

In this scenario, our search space for question planning is the following. First, we need to decide how many screens to show. Second, we need to determine what query properties our questions should focus on. Third, we need to decide how many answer options to display on each screen. Fourth, we need to pick those answer options.

We make those decisions based on a simple cost model, representing time overhead for crowd workers for verifying the current claim. We assume that workers read screen content from top to bottom. For each answer option, a worker needs to determine whether it is correct or not. We count a per-option verification cost in our model, distinguishing cost of verifying answers about query properties, v_p, from the cost of verifying the full query (on the final screen), v_f. We choose constants such that $v_p \ll v_f$ to account for the fact that full queries are significantly longer than their fragments (which increases reading time and therefore verification cost). If none of the given options applies, crowd workers must suggest an answer themselves. We denote by s_p and s_f the cost of suggesting answers for properties and queries (again, $s_p \ll s_f$).

First, we discuss how to choose the number of screens and answer options. We denote the number of screens by *nsc* and the number of options by *nop*. Predicting the precise verification cost for specific choices of those parameters is not possible. Doing so would require knowing the right solution to each question (as it determines how many options workers will read). However, we can upper-bound verification cost in relation to the cost of verifying claims without SCRUTINIZER (the proofs for this and the following theorems can be found in Appendix A.6).

Theorem 1. *Compared to the baseline, relative verification overhead of* SCRUTINIZER *is at most* $(nop \cdot v_f + nsc \cdot (v_p + s_p))/s_f$.

Corollary 1. *Setting* $nop = s_f/v_f$ *and* $nsc = s_f/(v_p + s_p)$ *limits verification overheads to factor three.*

We will use the aforementioned setting for most of our experiments. Having determined the number of screens and options, we still need to pick specific screens and answers. First, we discuss the selection of answer options. Note that the worst-case verification cost of a property depends only on the number of options shown (but not on the options themselves). Hence, to pick options, we consider expected verification cost instead.

We calculate expected verification cost based on our classifiers, assigning specific answer options to a probability. For a fixed property, denote by A the set of all relevant answer options. Also, denote by p_a the probability that an answer $a \in A$ is correct. We calculate expected verification cost when presenting users with an (ordered) list of answer options $\langle a_1, \ldots, a_m \rangle$ where $a_i \in A$.

Theorem 2. *The expected verification cost for answer options* $\langle a_1, \ldots, a_m \rangle$ *is* $v_p \cdot \sum_{i=1..m}(1 - \sum_{1 \leq j < i} p_{a_i})$.

Corollary 2. *Selecting answer options in decreasing order of probability minimizes expected verification cost.*

We illustrate screen design by an example.

Example 8. *Consider the query from Example 1. Our goal is to obtain feedback on the "Index" property. Our classifier ranks options in descending order of probability as follows: PGElecProd, PGElecDemand, and PGaccess. Assuming that it takes workers two times longer to look up the correct alternative themselves, compared to validating a given option, we select the first two options to display on screen (and a field to write the correct answer if not shown).*

Finally, we discuss the selection of query properties. Our goal is to select the best *nsc* properties to verify by creating corresponding screens. We define the quality of a property as follows. At any point, we consider a set of likely query translations for a claim. A good property has high pruning power with regards to the current set of candidates. This means that it allows us to discard as many incorrect candidates as possible. The following example illustrates pruning.

Example 9. *Assume a worker selects year 2030 in Figure 3.3. Implicitly, this decision prunes all possible expressions that use more (or less) than one symbol (to represent the year). E.g., we prune formula* `a+1` *since it contains too few placeholders.*

How many query candidates we can prune depends on the correct property value. Depending on the answer we obtain from the fact-checkers, more or less queries can be pruned. We do, of course, not know the correct answers when selecting questions. Hence, we define the expected pruning power of a set of properties as follows.

Definition 7. *Given a set Q of query candidates, a set S of query properties to verify, and trained models M predicting a-priori probabilities for possible answers, we define the pruning power $\mathcal{P}(S, Q, M)$ as the expected number of queries that are excluded by obtaining answers for S.*

Next, we provide a formula for pruning power, based on simplifying assumptions. For that, we denote by a_s^i the i-th answer option for property $s \in S$ and by $E_s^i \subseteq Q$ queries that are excluded if answer option a_s^i turns out to be correct. We assume independence between pruning probabilities for different query properties. While this assumption is simplifying, it allows us to use simple optimization algorithms (we may consider extensions in the future). Also, we assume that answer options are mutually exclusive (which holds for most, even not for all, query properties).

Theorem 3. *The pruning power $\mathcal{P}(S, Q, M)$ is given by $\sum_{q \in Q}(1 - \prod_{s \in S} \sum_{i: q \notin E_s^i} \Pr(a_s^i \text{ correct} | M))$.*

Next, we discuss the question of how to find property sets maximizing the above formula. Iterating over all possible property sets is possible, but expensive (exponential complexity in the number of properties). Instead, we select properties according to a simple, greedy approach. At each step, we add whichever property maximizes pruning

power to the set of selected properties (when comparing properties to add, we calculate pruning power for the union between the new and previously selected properties). We stop once the number of selected properties has reached the threshold determined before. An illustrative example follows.

Example 10. *We consider two queries and three properties. Assume property one prunes query one with probability 0.7 and query two with probability 0.3. The corresponding numbers are 0.5 and 0.9 for property two, and 0.1 and 0.95 for property three. We select screens for verifying two properties (i.e., nsc = 2). If selecting property one, the expected number of remaining queries is* $0.3 + 0.7 = 1$. *It is* $0.5 + 0.1 = 0.6$ *for property two and* $0.9 + 0.05 = 0.95$ *for property three. Hence, we greedily select property two first. Assuming independence, selecting property one next leads to an expected number of* $0.5 \cdot 0.3 + 0.1 \cdot 0.7 = 0.22$ *queries remaining. Selecting property three yields* $0.5 \cdot 0.9 + 0.1 \cdot 0.05 = 0.455$ *queries. Hence, we select property one next.*

While this algorithm may seem simple, it offers surprisingly strong formal guarantees. Those guarantees are derived from the fact that pruning power is a sub-modular function (Nemhauser and Wolsey, 1978). We define sub-modularity below.

Definition 8. *A set function* $f : S \mapsto \mathbb{R}$ *is sub-modular if, using* $\Delta_f(S, s) = f(S \cup \{s\}) - f(S)$*, it is* $\Delta_f(S_1, s) \geq \Delta_f(S_2, s)$ *for any* $S_1 \subseteq S_2$.

Intuitively, sub-modularity captures a "diminishing returns" behavior. If adding more elements to a set, the utility of new elements decreases as the set of previous elements grows. The pruning power function is sub-modular as well, according to the following theorem.

Theorem 4. *Pruning power is sub-modular.*

Next, we show that the simple greedy algorithm produces a near-optimal set of questions.

Theorem 5. *Using the greedy algorithm, we select a set of questions that achieve pruning power within factor* $1 - 1/e$ *of the optimum.*

Finally, we analyze time complexity (denoting by *nsc* the number of screens, by *npr* the number of properties, and by *nqu* the number of query candidates).

Theorem 6. *Finding optimal question sequences for verifying single claims is in* $O(nsc \cdot npr \cdot nqu)$.

3.5.2 Claim Ordering

We consider two criteria when selecting the next claims to verify. First, we consider the benefit of claim labels for training our classifiers (for automated claim to query translation). Second, we consider the expected verification cost.

The first point relates to prior work on active learning. Here, the goal is generally to select optimal training samples to increase the quality of a learned model. In our

case, verified claims correspond to training samples for classifiers that translate claims to queries. We follow the popular heuristic of picking training samples with maximal uncertainty and define the training utility as follows.

Definition 9. *Let $m \in M$ be a model predicting specific properties of the query associated with a text claim c. We assume that m maps each claim to a probability distribution over property values. Denote by $e(m, c)$ the entropy of that probability distribution. We define the training utility of c, $u(c)$ by averaging over all models (associated with different query properties): $u(c) = \sum_{m \in M} e(m, c)$.*

The second point (verification cost) relates to the cost model discussed in the previous subsection. However, this cost model is incomplete. It neglects the cost of understanding the context in which a certain claim is placed. Intuitively, verifying multiple claims in the same section is faster than verifying claims that are far apart in the input document. Our extended cost model takes this into account. It calculates verification cost for claim batches.

Definition 10. *Denote by C a batch of claims for verification. For each claim $c \in C$, denote by $s(c)$ the section in which this claim is located (instead of sections, a different granularity such as paragraphs can be chosen as well). Denote by $v(c)$ the pure claim verification cost for c defined in the last subsection. Further, denote by $r(s)$ the cost of reading (respectively skimming) section s. We define the total (combined verification and skimming) cost for claim batch C as the sum of both verification cost over all claims and reading cost over all associated sections: $t(C) = (\sum_{c \in C} v(c)) + (\sum_{s \in \{s(c) | c \in C\}} r(s))$.*

This cost model captures the desired property that verifying claims in the same section is faster. Our approach to claim ordering is based on this model. It is not useful to determine a global claim order before verification starts. We cannot predict how the quality of classifiers (and therefore claim verification cost) will change over time. Instead, we repeatedly select claim batches that are presented to the checkers. Those claim batches are selected based on training utility and the aforementioned cost model. To select claim batches, we solve the following optimization problem.

Definition 11. *Given a set of unverified claims C, the goal of claim selection is to select a claim batch $B \subseteq C$ such that total cost of B remains below a threshold t_m: $t(B) \leq t_m$. Additionally, the minimal and maximal batch size is restricted by parameters b_l and b_u: $b_l \leq |B| \leq b_u$. Under those constraints, the goal is to maximize accumulated training utility $\sum_{c \in B} u(c)$. Alternatively, as a variant, we minimize the cost formula $t(B) - w_u \cdot \sum_{c \in B} u(c)$ where w_u is a weight representing the relative importance of selecting claims with high uncertainty for classifier training.*

Analyzing complexity, we find the following.

Theorem 7. *Claim selection is NP-hard.*

The fact that claim selection is NP-hard justifies the use of sophisticated solver tools. We reduce the problem to integer linear programming. This allows us to apply mature

solvers for this standard problem. Next, we discuss how we transform claim selection into integer linear programming.

An integer linear program (ILP) is generally characterized by a set of integer variables, a set of linear constraints, and a (linear) objective function. The goal is to find an assignment from variables to values that minimizes the objective function, while satisfying all constraints.

We introduce binary decision variables of the form cs_i, indicating whether the i-th claim was selected ($cs_i = 1$) or not ($cs_i = 0$). Also, we introduce binary variables of the form sr_j to indicate whether section number j needs to be skimmed or not (to verify the selected claims). Next, we express the constraints of our scenario on those variables. First, we limit the number of selected claims to the range $[b_l, b_u]$ by introducing the linear constraints $b_l \leq \sum_i cs_i \leq b_u$. Next, we represent the constraint that sections of selected claims must be read. We introduce constraints of the form $sr_j \geq cs_i$ if claim i is located within section j. Furthermore, we limit accumulated verification cost of the selected claims by the constraint $(\sum_i cs_i \cdot v(c_i)) + (\sum_j sr_j \cdot r(s_j))$. Finally, we set $-\sum_i cs_i \cdot u(c_i)$ as objective function to minimize. We illustrate the transformation.

Example 11. *We select up to two out of three claims for verification (i.e., $b_u = 2$) with a time limit of 4 minutes. The first two claims are located within the same section, while the third one is in a separate section. We have uncertainties 0.3, 0.5, and 0.9 associated with the three claims. Each claim has an estimated verification time of one minute. We estimate one minute for skimming a section as well, and set w_u to one. We introduce decision variables cs_1 to cs_3, representing claim selections, and rs_1 and rs_2, representing skimmed sections. Verifying claims requires skimming their sections, therefore $cs_1 \geq sr_1$, $cs_2 \geq sr_1$, and $cs_3 \geq sc_2$. Under the constraint $\sum_{i=1..3} cs_i \leq 2$ and $(\sum_{i=1..3} cs_i) + sr_1 + sr_2 \leq 4$, we minimize $-0.3 \cdot cs_1 - 0.5 \cdot cs_2 - 0.9 \cdot cs_2$. The optimal solution selects claims two and three.*

The time complexity for solving a linear program generally depends on the solver and the algorithm it selects to solve a specific instance. However, the number of variables and constraints often correlates with solution time.

Theorem 8. *The size of the ILP problem is in $O(cc \cdot sc)$ where cc is the claim count and sc the section count.*

3.6 Experiments

We evaluated SCRUTINIZER using real data along four dimensions: (i) the accuracy and efficiency of the query generation from text w.r.t. state-of-the-art methods, (ii) the end-to-end effectiveness of the system in real verification tasks with domain experts, (iii) the effectiveness and efficiency of question scheduling, (iv) the impact of the quality of the user feedback on the results. The code of the system is available at https://github.com/MhmdSaiid/statchecker. More experimental results can be found in Appendix A.3.

Datasets. Our experiments are based on the two use cases described in Examples 3 and 4. For the coronavirus claims, we generated 3M true claims starting from the data (Appendix A.1). This synthetic corpus enabled us to bootstrap the classifiers, and we denote it as $\mathbf{C19_L}$. For testing the system with unseen claims, we analyzed the log of more than 30K claims tested by users on the website. We found that around 60% of the claims are statistical and, among those, we have the datasets to verify 70%. From the claims that the system can check, we manually annotated 55. For the IEA claims, we obtained a document of 661 pages, containing 7901 sentences, and the corresponding corpus of manually checked claims, with check annotations for every claim from three domain experts. The annotations cover 2053 numerical claims, out of which we identified 1539 having a formula that occurs at least five times in the corpus. We denote the resulting dataset as $\mathbf{IEA_L}$. After processing the claims, we identify 1791 relations, 830 row indexes, 87 columns, and 413 formulas. Around 50% of the values for all properties appear at most 10 times in the corpus, with the top 5% most frequent formulas appearing at least 8 times. In another paper (Saeed and Papotti, 2021), we further compare the systems with an additional dataset related to Basketball Data (BBL) (Reiter and Thomson, 2020)[1]. We utilize 132 tables with 1523 real annotated claims provided in the repository for the testing step. We generate ourselves the training data from the tables, as in the C19 scenario (Rezgui et al., 2021). We obtain an initial dataset of 32.3K samples, where 90% is used for training classifiers and 10% for validation. The dataset used for bootstrapping is denoted by $\mathbf{BBL_L}$ and the test dataset as $\mathbf{BBL_S}$. The datasets ($\mathbf{BBL_S}$ and $\mathbf{C19_S}$) are publicly available[2]. Examples datapoints of the datasets are shown below.

IEA Datapoint:

- **Claim**(underlined): Renewables and nuclear power generated **more** electricity than coal for the first time in 2019, and are on track to open a permanent lead.
- **Formula**: $(a + b) > c$
- **File**: outlookElectricity_World.csv (OEW)
- **Attribute Label**: 2019
- **Row Indices**: pgoutrenew (a), pgoutnuclear (b), pgoutcoal (c)
- **Query**:

```
q1 = SELECT 2019 FROM  OEW WHERE Index= pgoutrenew
q2 = SELECT 2019 FROM  OEW WHERE Index= pgoutnuclear
q3 = SELECT 2019 FROM  OEW WHERE Index= pgoutcoal
Final Query = SELECT ((q1 + q2) > q3)
```

[1] https://github.com/ehudreiter/accuracySharedTask
[2] https://zenodo.org/record/5128604#.YPrSgXUzZuU

C19 Datapoint:
- **Claim**: The number of confirmed cases increased in France in from March 2022 to May 2022.
- **Formula**: $b > a$
- **File**: ConfirmedCases.csv
- **Attribute Labels**: March_2022, May_2022
- **Row Index**: France

BBL Datapoint:
- **Claim**: Mike Conely scored the most points in the Utah Jazz vs Boston Celtics game.
- **Formula**: $Max(Points) == a$
- **File**: UtahBoston.csv
- **Attribute Label**: Points
- **Row Index**: Mike Conely

3.6.1 Statistical Claim to Query

We compare the performance of our query generator solution against three state-of-the-art systems. We characterize such systems by proposed dimensions, on the input and output levels, as a basis for comparison (Saeed and Papotti, 2021). The dimensions can be found in Appendix A.5.

The first system, AggChecker (Jo et al., 2019), translates statistical claims into SQL queries for verification. The second, TabFact, exploits PLMs to encode the linearized table (taken as input) and a statement into continuous vectors for verification (Chen et al., 2020b). We fine-tuned the PLM with the training examples in our experiments. The third, Tapas, is a question answering system that extends BERT's architecture to encode tables as input (Herzig et al., 2020). Its pre-trained semantic parsing model takes as input a sentence and a table obtaining state-of-the-art accuracy in several datasets. We tried automatically translating claims to questions as pioneered by the ClaimBuster system (Hassan et al., 2017a). The precision was however not satisfactory (e.g., we did not obtain any questions for 7 out of the 20 IEA claims considered next), upper-bounding the precision of even a perfect natural language query interface. Instead, we manually translated claims into questions. For $\mathbf{BBL_S}$, we relied on a pattern-based script to generate questions.

Both natural language baselines have limitations on the input data and on the query space. However, they report top performance in semantic parsing datasets because the majority of examples in these are limited to one (small) relation and simple operations (Chen et al., 2020b; Herzig et al., 2020). These limitations are reflected in the smaller percentage of claims that the baselines can handle with their set of functions, as reported in Table 3.2. One of the reasons why Scrutinizer covers a much larger number of claims is its ability to combine multiple functions in the verification expression, however, the system is limited by the amount of available data. More than 22% of the

Table 3.2: Ratio of supported claims.

	AggChecker	Tapas	TabFact	Scrutinizer
IEA$_L$	33.06%	17.02%	27.28%	74.96%
C19$_L$	21.49%	21.49%	37.82%	100%
BBL$_L$	53.00 %	56.17%	56.17%	56.17%

Table 3.3: Verification accuracy on the small datasets.

	Tapas	TabFact	AggChecker	Scrutinizer
C19$_S$	0.64	0.76	0.44	0.80
IEA$_S$	0.07	0.58	0.5	0.65
BBL$_S$	0.41	0.17	0.13	0.51

claims in **IEA$_L$** has a total of three or more variables and functions (42% for log claims in **C19$_L$**), we observed expressions with up to 14 operators. To enable a comparison against the baselines, we selected a subset of suitable claims from our datasets. We automatically identified explicit claims that require only one relation for the verification and with mathematical operations supported by both baselines. We denote these simpler datasets as **C19$_S$** and **IEA$_S$**. After the selection, these small datasets contain 0.92% and 15.6% of the claims in the original **C19$_L$** and **IEA$_L$**, respectively. As indicated for **BBL$_S$**, we utilize the claims in the repository and generate data for **BBL$_L$**.

All baseline methods require the associated relation as input, so we fed the same information also to Scrutinizer. For Tapas, we limit the input to a sample of 11 tuples, including the one needed to verify the claim, as the system fails with the entire relation as input. For Scrutinizer and TabFact, the training set contains 3000 claims, while Tapas and AggChecker have no training step. Notice that for Scrutinizer we use the top-10 output from the classifiers without relying on the user feedback in this experiment. We also note that our system requires having the SQL template as an input, since it follows a slot-filling approach to generate the final query (Choi et al., 2021).

Table 3.3 reports the results of the experiments with the four systems on the test claims filtered from the large datasets to be supported by the baselines.We observe that Scrutinizer outperforms all the baselines. Moreover, only AggChecker and Scrutinizer return a query to "explain" their decision, while the others are not immediately interpretable. It is also evident that Coronavirus claims are easier to handle as the number of operations, rows and attributes is smaller than for the IEA claims. For **C19$_S$**, Scrutinizer fails only for claims which require formulas that it has not learned yet. For **BBL$_S$**, low results are explained by the fact that all systems have low coverage of the claims in the data and some claims require verification that spans across multiple tables. We note that Scrutinizer requires annotations on multiple levels (table, row index,

Table 3.4: Execution time of the test datasets (seconds).

	AggCheck	Tapas	TabFact	Scrutinizer
C19$_S$	280.41	991.52	23.09	0.03
IEA$_S$	321.53	943.25	18.11	0.68
BBL$_S$	3472.44	12709.45	40.20	236.64

attribute label, and formula), making it more 'label-demanding' than other systems.

For the execution times, we distinguish the training and the testing. Classifier-training time is needed for Scrutinizer and TabFact; however, this is typically negligible (on the studied datasets) with the usage of GPUs. AggCheck and TabFact, on the other hand, have zero setup time. We report the execution times for all test data in Table 3.4. Tapas is the slowest as the model is jointly computing the relevant cells and performing an operation on them, compared to TabFact that requires a negligible amount of time to perform binary classification. Scrutinizer consumes negligible time in classifier predictions, but the brute-force query generation process could potentially take considerable amount of time when multiple combinations are available. AggCheck, although having to perform evaluations of a large number of queries, successfully merges the execution of similar queries to increase efficiency. In summary, all systems are usable in reasonable time in our experience in an entry-level infrastructure with a low-end GPU.

3.6.2 User Study

In this experiment, we involved seven domain experts from the institution to measure the benefit of our system compared to the traditional manual workflow for verification. We trained Scrutinizer with all the annotated statistical (real world) claims and randomly selected 43 claims among the ones with the 10 formulas that cover the majority of the claims. As we only have access to the correct version of the claim, we randomly selected 25% of them to inject errors.

Three experts have been randomly assigned to the **Manual** process and the remaining four to the **System**-assisted process. We gave them instructions to execute the test without interruptions and without collaboration. Three claims (two correct, one incorrect) have been used for training on the new process and the remaining 40 for the study. The task given to the experts was to verify as many claims as possible in 20 minutes, given access to their traditional tools in the manual process (spreadsheets and databases) and to our system only in the second case. The order of the claims has been fixed to allow comparison among experts, and the time for checking every claim has been registered.

We distinguish three cases: skipped claims, claims that have been correctly labeled, and incorrect decisions. Results for each checker are reported in Figure 3.4. Considering correct and incorrect checks, on average a user verifies 7 claims manually and 23 claims with Scrutinizer in 20 minutes. Users tend to skip a comparable amount of claims in both settings. In the System process, a few claims have been incorrectly checked. Those

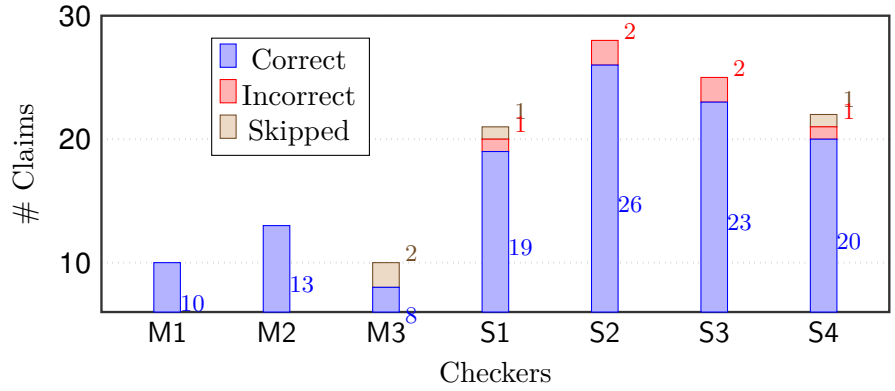

Figure 3.4: Number of claims verified in 20 minutes by checkers with the manual process (M1–M3) and with our system (S1–S4).

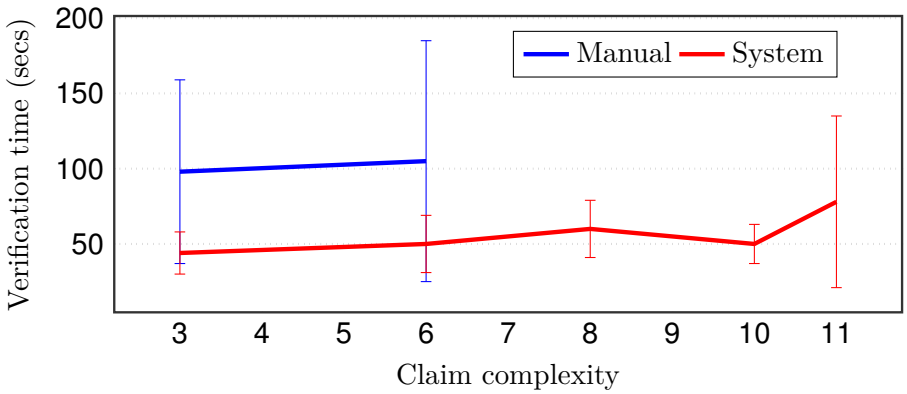

Figure 3.5: Average time to verify claims of increasing complexity with the Manual and System processes.

are all correct claims labeled as incorrect. However, by using simple majority voting over three checkers, the accuracy of the aggregate answers is 100% for both the Manual and the System groups. There was only one claim where verification time using the tool surpassed the traditional manual verification time. After investigating this case, it turned out that this was due to sequential checking. The user consulted a relation different from what we were expecting him to choose. The different relation led him to the correct answer, but such relation was used also in the previous question with the same primary key and attributes values, making this claim very fast to verify.

We also report in Figure 3.5 average verification time and standard deviation for the two groups of checkers with claims of increasing complexity. The claim complexity is the sum of the elements in the query to verify it: number of key values, attributes, operations, constants and variables. Checkers using Scrutinizer take on average less than half the time to verify claims of the same complexity. The average time taken

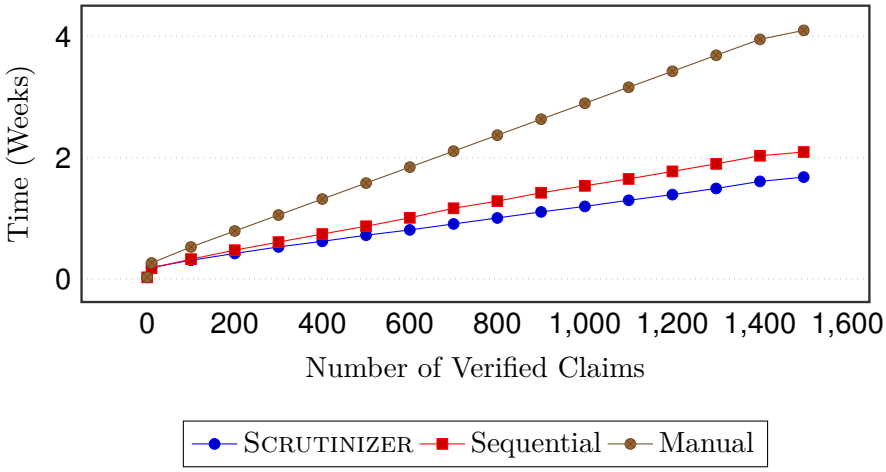

Figure 3.6: Accumulated verification time over verification period.

by checkers using our system for claims of 11 elements is lower than the time used by checked with the manual process with 6 elements. We report in the plot claims for which at least two checkers have been able to process it. We are therefore not showing in the plot a checker using Scrutinizer who took on average 29 seconds to verify two claims of complexity 14. We remark that for one claim of complexity 6, it took 203 seconds for one of the Manual users to verify it, while for the same claim the slowest System user spent 66 seconds on the same task.

We conducted the study on a laptop (1.80GHz x 8 i7 CPU, 32 GB of memory). For any claim, testing a classifier took less than 0.2 seconds and query generation took less than half a second (0.35 seconds on average).

3.6.3 Simulation

In the previous subsection, we have demonstrated that Scrutinizer decreases verification overheads for single claim batches. Next, we study the efficiency of Scrutinizer when verifying entire reports. Verification time for entire reports is typically in the order of months for IEA. Hence, we cannot use another user study. Instead, we created a simulator, based on the results of our initial user study. We simulate the verification of the 2018 IEA world energy outlook report, using the original claims and original data. We assume a team of three fact-checkers (which is typical for IEA). We simulate a "cold start" scenario, meaning that our classifiers have no initial training data. Instead, they use claim labels provided by simulated fact-checkers. This corresponds to the worst case for our system. It represents a scenario in which the very first version of a new report is received and verified. Our model for verification time per claim is based on time measured in the user study. It takes into account reduced verification overheads once proposed query fragments are accurate. We compare three baselines. First, we consider manual verification ("Manual") which is the current default. Each claim is verified without any computational support. Second, we consider a simplified version of Scrutinizer. This

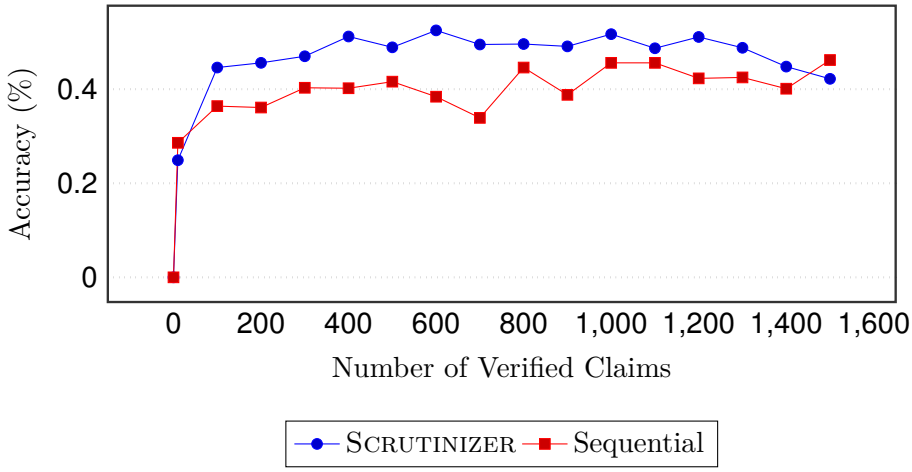

Figure 3.7: Evolution of Scrutinizer and sequential average accuracy over verification period.

version ("Sequential") does not optimally reorder claims, as described in Section 3.5.2, but verifies them sequentially (i.e., in document order) instead. We compare those two approaches against the Scrutinizer system. For Sequential and Scrutinizer, we assume that ten answer options are shown per property. For Scrutinizer, we use claim batches of size 100, after which we retrain classifiers and select the next claims to verify via ILP. Our simulator is implemented in Python 3, using Gurobi 9.0.1 as ILP solver. Experiments were executed on a MacBook Pro with 2.4 GHz Intel Core i5 processor and 8 GB memory.

Table 3.5: Summary of simulation results.

	Manual	**Sequential**	**Scrutinizer**
Time (Weeks)	4.1	2.1	1.7
% Savings	-	49%	59%
Avg. Accuracy	-	40%	47%
Max Accuracy	-	46%	53%
Comp. (Mins)	-	14	28

Table 3.5 summarizes simulation results. We report total verification time for all three fact-checkers, assuming an eight-hour work day and a five-day week. We make the following observations. First, using Scrutinizer reduces verification overheads by more than factor two (circa 60%). This is consistent with the results of our user study. At the same time, it is remarkable since we consider a cold start scenario. The results show that, given a sufficiently large document to verify, the initial warm-up period of the classifiers does not impact overall performance by too much. Second, we observe a

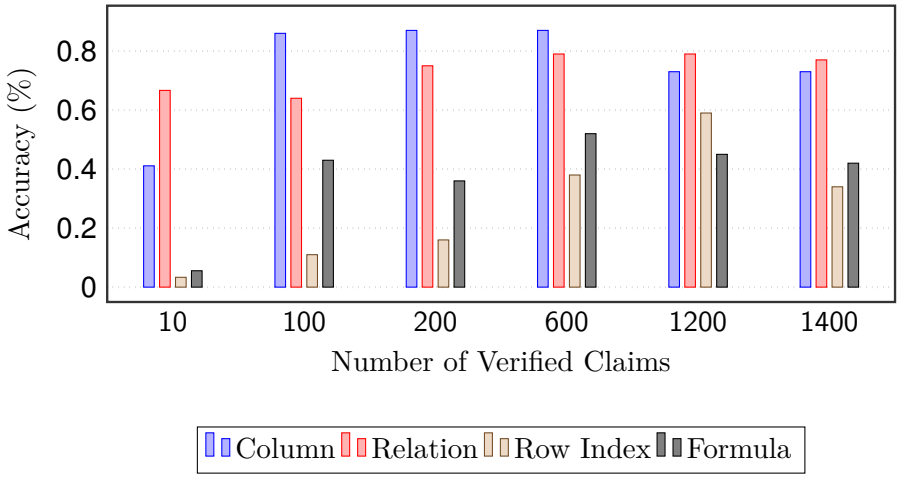

Figure 3.8: Evolution of classifier accuracy over verification period.

positive impact due to claim ordering. While using SCRUTINIZER without that feature is still helpful, cost savings increase when claims are systematically prioritized. Table 3.5 shows that, in the latter case, the average (and maximal) quality of classification over the entire period improves. Figure 3.6 shows that SCRUTINIZER and the sequential baseline are near-equivalent at the beginning of the classification process. Claim ordering pays off more and more as verification proceeds. At the same time, computational overheads are negligible for all compared systems. SCRUTINIZER spends 15 minutes in total to plan optimal question sequences, and for selecting optimal claims via ILP. The remaining 13 minutes are due to retraining classifiers.

Figure 3.7 analyzes classifier accuracy as a function of verification time. SCRUTINIZER has initially a lower accuracy, as it selects claims that have low classifier confidence (therefore useful for learning) but are relatively cheap to verify. Soon, active learning starts paying off and the accuracy of SCRUTINIZER dominates the baseline. SCRUTINIZER postpones verifying particularly expensive claims that are associated with low classifier confidence. Once running out of other options, those claims are verified at the very end (leading to a drop in accuracy).

Figure 3.8 decomposes accuracy for SCRUTINIZER (with claim ordering) according to classifier type. The effect discussed in the last paragraph (a steep increase followed by a drop towards the end) still hold when considering classifiers separately. Further, we notice that certain properties are harder to infer from text. For instance, inferring row indices is among the hardest classification tasks. This is intuitive, as the classification domain (i.e., number of rows) is typically larger than for other classifiers (e.g., columns).

Finally, in Figure 3.9, we analyze accuracy for the top-k labels and for different classifiers. In most cases, classifiers reach most of their potential with the first 10 entries.

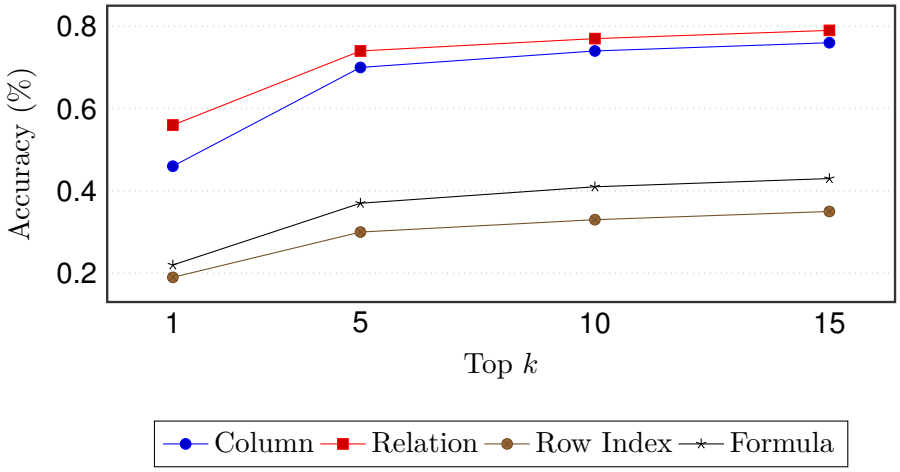

Figure 3.9: Top k accuracy for different SCRUTINIZER classifiers as a function of k.

3.6.4 Quality of User Feedback

We measure the impact of the quality of the annotations in three experiments: by injecting synthetic errors, by varying the size of the training data, and by training the models with the noisy examples gathered from the Web users of the system. We then repeat each experiment three times and report the average accuracy. We report the results for $k = 5$, as this is the default value in our experiments.

We start with the study of how the label error rate in the $\mathbf{IEA_L}$ training data affects the classifier accuracy. For each property, we randomly split the annotated claims into training and test sets (0.9/0.1 ratio). We then select a percentage of training claims at random and change their target label with one of the possible labels (minus the correct one). Figure 3.10 shows how accuracy varies as a function of the error rate for top-5 predictions. With an increasing number of mislabeled data points, all classifiers show decreasing accuracy. However, even with up to 30% error rate, the results are close to the case without noise. In our experiments, IEA checkers error rate is below 10% and even Website users do not exceed the 30% rate. Finally, as expected, the accuracy gets close to random guessing for very high ratio of error injection (0.9) and classifiers with a small number of possible outputs perform better at any rate. We also observe similar patterns with top-1 results.

In a second experiment, we vary the train/test ratio by sampling increasing portions from the dataset and testing on the others. Figure 3.11 shows how properties with high number of classes (e.g., row_index) perform poorly when the training data is small compared to properties with a low number of classes (e.g., column).

Finally, we report on an experiment conducted with classifiers trained with the 1388 answers collected from our Web interface for the test dataset of $\mathbf{C19_L}$. Table 3.6 shows that, even in a very noisy setting, without majority voting to filter out incorrect answers, the classifiers for Relation and Row Index perform well, but accuracy for the Column classifier is very low. In $\mathbf{C19_L}$, the number of classes when classifying relations, rows,

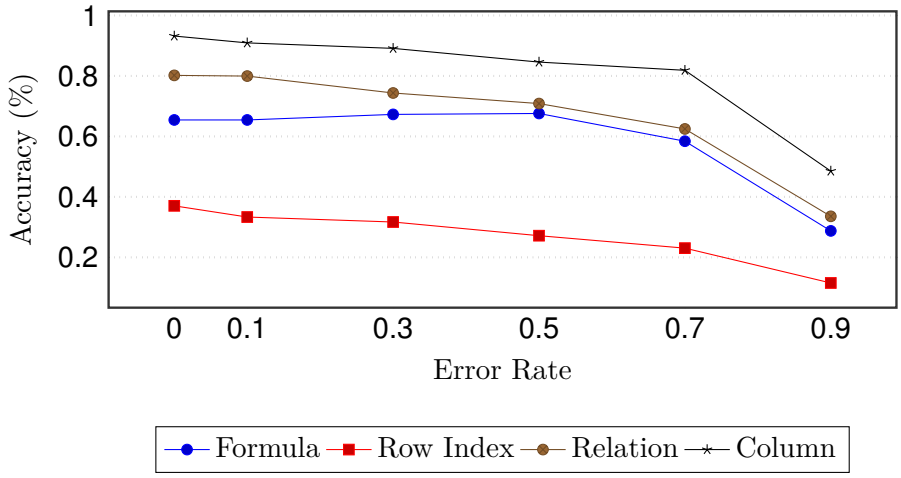

Figure 3.10: Classifier accuracy as a function of the percentage of erroneous training claims.

Table 3.6: Accuracy on $\mathbf{C19_L}$ vs. training data quality.

	Relation	Column	Row Index
Web crowd	0.84	0.04	0.74
Generated	1	0.98	0.96

and columns, are all comparable. However, the semantics of claim text fragments tends to be time-sensitive for the column classifier alone (e.g., semantics of the expression "this month"). If re-training the column classifier each month (using labels collected in that month alone), accuracy varies between 0.12 to 0.49 for the months January to June (Appendix A.3.4). While we did not collect user answers for the formula, we manually annotated 40 real examples from the log for the Formula classifier. We picked two formulas that were not in the synthetic training data, i.e., $max(a)$ and $max(a/b)$, with 20 labels each. We then incrementally added them to the training data and tested on the remaining claims. The results show that the system is able to learn the two new formulas after 12 examples per class. Also, in a small experiment described in more detail in Appendix A.3.5, we verified that classifier confidence is significantly higher for claims translating to queries that match the current template (confidence of 0.39 versus 0.94).

3.7 Related Work

Scrutinizer targets the verification of numerical claims from raw data. Our scope differs from prior work not focused on verification (e.g., work on identifying check-worthy claims (Hassan et al., 2017a; Jaradat et al., 2018) or on studying misinformation spread (Sherchan et al., 2013; Ferrara et al., 2016)), from prior work verifying claims

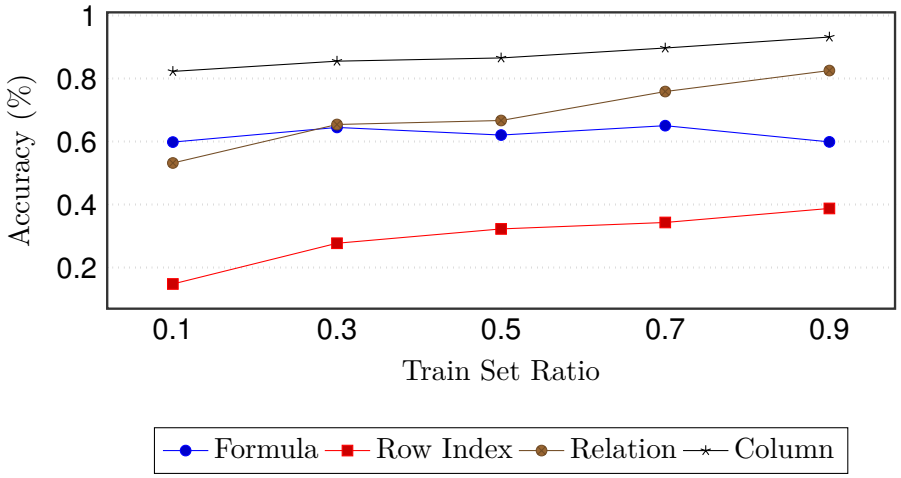

Figure 3.11: Accuracy versus training set size.

given as logical formulae (e.g., for verifying claim robustness (Wu et al., 2014)), from work on verifying claims from different types of data (e.g., Web documents (Mihaylova et al., 2018; Thorne and Vlachos, 2018; Wang et al., 2018b) or databases of previously checked claims (Karagiannis et al., 2019; Hassan et al., 2017b)), and from work that focused on different claim types (e.g., claims associating entities with non-numerical properties (Ciampaglia et al., 2015; Shi and Weninger, 2016; Gardner et al., 2014; Gardner and Mitchell, 2015; Huynh and Papotti, 2019)). Scrutinizer connects to work on explainable fact-checking (Leblay, 2017; Gad-Elrab et al., 2019; Ahmadi et al., 2019) as it verifies claims via queries whose structure can be explained to users.

Scrutinizer relates to prior efforts on data-driven analysis of statistical claims. BriQ can map one explicit claim to the input dataset and supports only 6 operations for a single user (Ibrahim et al., 2019). StatSearch considers a corpus of relations, but supports only SP queries (no functions) and can search (not verify) one explicit claim with a single user (Cao et al., 2018). The closest work is the AggChecker system (Jo et al., 2019) in that it translates statistical claims into SQL queries for verification. Scrutinizer supports however a richer query model (see Section 3.6), it assumes verification by a crowd of fact-checkers (instead of single users), and it supports verification of long documents (as opposed to short texts) by features such as question reordering for active learning. Prior work on mixed-initiative fact-checking (Gatterbauer et al., 2009) does not translate claims to SQL queries.

We translate text to SQL queries, thereby connecting to prior work on natural language query interfaces (NLQI) (Agrawal et al., 2002; Li and Jagadish, 2014; Zhong et al., 2017; Li and Jagadish, 2016; Saha et al., 2016; Weir et al., 2019; Iyer et al., 2017) and learning query interfaces (Iyer et al., 2017; McCamish et al., 2019) in general. Our scenario differs in multiple ways. First, most prior work assumes that domain-specific training data is either already available (Zhong et al., 2017) or not required (Li and Jagadish, 2016, 2014; Agrawal et al., 2002). We propose methods for efficiently acquiring domain-specific

training data and show experimentally that generic methods do not work for our use cases (Chen et al., 2020b; Herzig et al., 2020). Second, prior work (Agrawal et al., 2002; Li and Jagadish, 2014; Zhong et al., 2017; Li and Jagadish, 2016; Saha et al., 2016; Weir et al., 2019; Iyer et al., 2017; McCamish et al., 2019) typically considers translation of single queries. Here, we consider large claim collections. This motivates claim context as a feature (see Section 3.4) or claim selection as a planning problem (see Section 3.5). We do not reason over the database with neural networks as other work (Thorne et al., 2021), but instead use classifiers to predict query properties, thus offering explainability of the system's decision. Finally, unlike prior work (Iyer et al., 2017), we decompose query translation into sequences of simple questions (see Section 3.5.1) as writing entire SQL queries is beyond the capabilities of domain experts employed for verification.

3.8 Conclusion

We introduced Scrutinizer, the first system for crowd-sourcing the verification of general statistical claims. Our solution effectively minimizes the amount of work needed by a group of domain experts to verify textual claims in a document. We find that professional fact-checkers from IEA verify claims twice as fast using Scrutinizer, compared to their traditional workflow. However, this chapter covers only one of the two claims shown in Figure 1.1. The first claim requires reasoning over facts and rules, which Scrutinizer is not capable of. That said, we will see in the next chapter how to (i) augment PLMs with soft logic rules, and (ii) test their logical performance on external datasets.

4 Emulating Soft Reasoners with RuleBert

While PLMs are the go-to solution to tackle many natural language processing problems, they are still very limited in their ability to capture and to use common-sense knowledge. In fact, even if information is available in the form of approximate (soft) logical rules, it is not clear how to transfer it to a PLM in order to improve its performance for deductive reasoning tasks. Verifying the first claim in Figure 1.1, requires the PLM to reason over rules and facts. Not only that, but as some rules are accompanied by weights expressing their confidence, it is key to incorporate such weights during the reasoning process in order to verify a certain claim. In this chapter, we aim to bridge this gap by teaching PLMs how to emulate reasoning with soft Horn rules. We introduce a classification task where, given facts and soft rules, the PLM should return a prediction with a probability for a given hypothesis. We release the first dataset for this task, and we propose a revised loss function that enables the PLM to learn how to predict precise probabilities for the task. Our evaluation results show that the resulting fine-tuned models achieve very high performance, even on logical rules that were unseen at training. Moreover, we demonstrate that logical notions expressed by the rules are transferred to the fine-tuned model, yielding improvements on external datasets.

4.1 Introduction

PLMs based on transformers (Devlin et al., 2019; Liu et al., 2020b) are established tools for capturing both linguistic and factual knowledge (Clark et al., 2019b; Rogers et al., 2020). However, even the largest models fail on basic reasoning tasks. If we consider common relations between entities, we see that such models are not aware of negation, inversion (e.g., *parent-child*), symmetry (e.g., *spouse*), implication, and composition (Kassner et al., 2020). While these are obvious to a human, they are challenging to learn from text corpora as they go beyond linguistic and factual knowledge (Ribeiro et al., 2020; Kassner and Schütze, 2020). We claim that such reasoning primitives can be transferred to the PLMs by leveraging logical rules, such as those shown in Figure 4.1. Such augmented

Input facts:
Mike is the parent of **Anne**. **Anne** lives with Mark. **Anne** is the child of Laure. **Anne** lives with *Mike*.
Input rules:
(r_1, .1) Two persons living together are married.
(r_2, .7) Persons with a common child are married.
(r_3, .9) Someone cannot be married to his/her child.
(r_4, 1) Every person is the parent of his/her child.

Test 1: Laure and *Mike* are married.
Answer: *True* with probability 0.7 $[r_4, r_2]$

Test 2: **Anne** and Mark are married.
Answer: *False* with probability 0.9 $[r_1]$

Test 3: **Anne** and *Mike* are married.
Answer: *False* with probability 0.9 $[r_1, r_3, r_4]$

Figure 4.1: Examples of hypotheses that require reasoning using facts and possibly conflicting soft rules (confidence shown to the right, with rule ids shown in brackets).

PLMs can aid in the verdict prediction phase of a fact-checking system (Nakov et al., 2021a), resolving some of the logical deficiencies of standard PLMs.

While there have been initial attempts to teach reasoning with rules to PLMs (Clark et al., 2020; Kassner et al., 2020), such approaches model only a subclass of logical rules. In fact, current solutions focus on exact rules, i.e., rules that hold in all cases. In reality, most of the rules are approximate, or soft, and thus have a certain confidence of being correct. For example, across the 7,015 logical rules defined on the DBpedia knowledge graph, only 11% have a confidence above 95%. In the example, rules r_1–r_3 are soft, i.e., cover knowledge that is not true in all circumstances. Consider rule r_2, stating that if two persons have a child in common, they are most likely married. As r_2 has a confidence of being correct of 0.7, this uncertainty is reflected in the probability of the prediction.

With the above considerations in mind, here we show how to reason over soft logical rules with PLMs. We provide facts and rules expressed in natural language, and we ask the PLM to come up with a logical conclusion for a hypothesis, together with the probability for it being true.

Unlike previous approaches (Clark et al., 2020), we enable deductive reasoning for a large class of soft rules with binary predicates and an unrestricted number of variables. Our model can even reason over settings with conflicting evidence, as shown in Test 3 in Figure 4.1: In the example, as Anne and Mike live together, they have a 0.1 confidence of being married because of soft rule r_1. However, we can derive from exact rule r_4 that Anne is the child of Mike, and therefore they cannot be married, according to soft rule r_3. Without rule confidences, reasoning in such conflicting setting would be impossible.

To model uncertainty, we pick one flavor of probabilistic logic programming languages, LP^{MLN}, for reasoning with soft rules (Lee and Wang, 2016). It assigns weights to stable models, similarly to how Markov Logic assigns weights to models. However, our method

is independent of the logic programming approach at hand, and different models can be fine-tuned with different programming solutions. Our proposal makes use of synthetic examples that "teach" the desired formal behavior through fine-tuning. In particular, we express the uncertainty in the loss function used for fine-tuning by explicitly mimicking the results for the same problem modeled with LP^{MLN}.

Our contributions in this chapter can be summarized as follows:

- We introduce the problem of teaching soft rules expressed in a synthetic language to PLMs through fine-tuning (modeled as binary classification). This is understudied in the claim verification phase of a fact-checking system, as usually hard rules are addressed.
- We create and release the first dataset for this task, which contains 3.2M examples derived from 161 rules describing real common-sense patterns with the target probability for the task obtained from a formal reasoner (Section 4.4).
- We introduce techniques to predict the correct probability of the reasoning output for the given soft rules and facts. Our solution relies on a revised loss function that effectively models the uncertainty of the rules (Section 4.5). Our approach handles multi-variable rules and nicely extends to examples that require reasoning over multiple input rules.
- We show that our approach enables fine-tuned models to yield prediction probability very close to that produced by a formal reasoner (Section 4.6). Our PLM fine-tuned on soft rules, RuleBERT, can effectively reason with facts and rules that it has not seen at training, even when fine-tuned with only 20 rules.
- We demonstrate that our fine-tuning approach effectively transfers knowledge about predicate negation and symmetry to the lower levels of the transformer, which benefits from the logical notions in the rules. In particular, RuleBERT brings enhancement in performance on three external datasets.

The data, the code, and the fine-tuned model are available at http://github.com/MhmdSaiid/RuleBert.

4.2 Related Work

PLMs have been shown to portray some reasoning capabilities (Talmor et al., 2020b), but fail on basic reasoning tasks (Talmor et al., 2020a; Helwe et al., 2021) and are inconsistent (Elazar et al., 2021), especially when it comes to negation (Kassner and Schütze, 2020). While some inconsistencies could be addressed by further training (Hosseini et al., 2021), our work focuses on deductive reasoning for PLMs, where alleviating logical inconsistency is a side-product. Note that our work is different from previous work, e.g., on measuring the factual knowledge of PLMs (Petroni et al., 2019), on probing the commonsense capabilities of PLMs at the token or at the sentence level (Zhou et al., 2020b), or on testing the reasoning capabilities of PLMs on tasks such as age comparison and taxonomy conjunction (Talmor et al., 2020a). Our work relates to Task #15 in the bAbI dataset (Weston et al., 2016) and to RuleTakers (Clark et al., 2020). However,

we differ (*i*) by using a larger subclass of first-order logic rules (with more variables and various forms), and (*ii*) by incorporating soft rules. Our proposal is different from work on Question Answering (QA) with implicit reasoning based on common-sense knowledge (Clark et al., 2019a), as we rely purely on deductive logic from explicitly stated rules. Our approach also differs from methods that semantically parse natural language into a formal representation on which a formal reasoner can be applied (Liang, 2016), as we directly reason with language. Unlike previous work, we do not design a new, ad-hoc module for neural reasoning (Yang et al., 2017; Hamilton et al., 2018; Minervini et al., 2020), nor integrate explicit symbolic reasoners (Zhang et al., 2022), but rely solely on the transformer's capability to emulate algorithms (Wang et al., 2019; Lample and Charton, 2020). Our work is related to Natural Language Inference (NLI) and textual entailment, but we deal with soft Horn rules expresses in language (MacCartney and Manning, 2009; Dagan et al., 2013). Our work is related to the reasoner module of our system ExpClaim (Ahmadi et al., 2019). The reasoner module takes similar input to that of RuleBERT, but in a logical form expressed in a probabilistic logical program. With RuleBERT, we emulate the reasoner module of ExpClaim with input expressed in synthetic English.

4.3 Background

In this section, we discuss logical rules that are obtained from DBpedia (Section 4.3.1), how their confidences are computed (Section 4.3.2), and one flavor of soft logic that is used in our experiments (Section 4.3.3).

4.3.1 Logical Rules

We rely on existing corpora of declarative *Horn rules* mined from large RDF KGs (Galárraga et al., 2015; Ortona et al., 2018a; Ahmadi et al., 2020). An atom in a rule is a predicate connecting two universally quantified variables. A Horn rule (or clause) has the form: $B \rightarrow h(x, y)$, where $h(x, y)$ is a single atom (head or conclusion of the rule) and B (body or premise of the rule) is a conjunction of atoms. Positive rules identify relationships between entities, e.g., r_1, r_2, r_4 in Figure 4.1. Negative rules identify contradictions, e.g., r_3 in Figure 4.1. Rules can contain predicates comparing numerical values, such as $<$. For example, negative rule r_5: *birthYear(b,d)* $\wedge$ *foundYear(a,c)* $\wedge$ *<(c,d)* $\rightarrow$ *negfounder(a,b)* states that any person (variable b) with a birth year (d) higher than the founding year (c) of a company (a) cannot be its founder. A fact is derived from a rule if all the variables in the rule body are replaced with constants from facts. For r_5, facts "foundYear(Ford,1903), birthYear(E. Musk,1971), >(1971,1903)" trigger the rule that derives the fact *negFounder*(E. Musk, Ford).

4.3.2 Rule Confidence

Exact rules, such as r_4, apply in all cases, without exception. However, most rules are approximate, or soft, as they apply with a certain likelihood. For example, r_3 in

Figure 4.1 is true in most cases, but there are historical exceptions in royal families. Rules are annotated with a measure of this likelihood, either manually or with a computed *confidence*. To compute rule confidences, we start with the notion of confidence of a positive rule from the literature (Galárraga et al., 2015).

The *support* of a rule is defined as the number of distinct pair of subjects and objects in the head of all instantiations of the rule that appear in the KG:

$$supp(\vec{B} \rightarrow h(x,y)) := \#(x,y) : \exists z_1, ..., z_n : \vec{B} \wedge h(x,y) \tag{4.1}$$

where $z_1, ..., z_n$ are the non-target variables.

We remark that the support makes use of the non-target variables, but counts only the number of distinct pairs for the head values. Consider the rule "if two persons have a child in common, they are in the spouse relation" ($hasChild(a,c) \wedge hasChild(b,c) \rightarrow spouse(a,b)$) and the Smith family. Assume Will Smith, Jada Pinkett, and their two children are in the KG with the correct Spouse and Child triples to represent their relationships. The support would count this family as one occurrence. While it uses the non-target variables (referring to the child), the measure does not count this family twice, despite the rule can be instantiated twice (once for each child). This design choice takes care of possible skews in the data, making sure that a rule does not get assigned a very high support when it applies to only one (distinct) pair of head entities.

The *counter-support* of a rule quantifies the number of false predictions over the existing KG. A challenge to compute this number is that KGs do not provide negative evidence, but we want to claim a mistake only if some evidence to support the case exists. To address this issue, we rely on a Local Closed World Assumption. This assumption states that if we know one y (resp. x) for a given x (resp. y) and h, then we know all y (resp. x) for that x (resp. y) and h. This is widely used in practice and has proven to be an effective heuristic to overcome incompleteness of KGs (Galárraga et al., 2015; Dong et al., 2014; Ortona et al., 2018).

$$counter\_supp(\vec{B} \rightarrow h(x,y)) := \#(x,y) : \exists z_1, ..., z_n, h(x,y') \vee h(x',y) : \vec{B} \wedge \neg h(x,y) \tag{4.2}$$

For example, a rule predicts that "Luke" and "Mary" are married, this triple is not in the KG, but "Luke" is already reported as married to someone else in the KG. To take into account both true and false predictions for a rule, we introduce *confidence scores* for positive and negative rules.

Positive Rules. Considering a positive rule $\vec{B} \rightarrow h(x,y)$ for relation $h(x,y)$. We define its confidence score as following:

$$conf(\vec{B} \rightarrow h(x,y)) := \frac{supp(\vec{B} \rightarrow h(x,y))}{supp(\vec{B} \rightarrow h(x,y)) + counter\_supp(\vec{B} \rightarrow h(x,y))} \tag{4.3}$$

Negative Rules. Consider a negative rule $\vec{B'} \rightarrow \neg h(x,y)$ for relation $h(x,y)$. We define

its confidence score as following:

$$conf(\vec{B'} \rightarrow \neg h(x,y)) := \frac{counter\_supp(\vec{B'} \rightarrow h(x,y))}{counter\_supp(\vec{B'} \rightarrow h(x,y)) + supp(\vec{B'} \rightarrow h(x,y))} \tag{4.4}$$

Intuitively, by definition, the support (resp. counter-support) of a negative rule $\vec{B'} \rightarrow \neg h(x,y)$ is indeed the counter-support (resp. support) of the corresponding positive rule $\vec{B'} \rightarrow h(x,y)$.

4.3.3 Probabilistic Answer Set Programming

As we deal with soft rules, we adopt LP$^{\text{MLN}}$ (Lee and Wang, 2016) to create the dataset. LP$^{\text{MLN}}$ is a probabilistic extension of answer set programs (ASP) with the concept of weighted rules from Markov Logic (Baral, 2010). In ASP, search problems are reduced to computing *stable models* (answer sets), a set of beliefs described by the program. A weight (or confidence) is assigned to each rule, so that the more rules a stable model satisfies, the larger weight it gets, and the probability of the stable model is computed by normalizing its weight among all stable models. Given a set of soft rules and facts, we measure how much the hypothesis is supported by the stable model. More details can be found in Appendix B.1.

4.4 Dataset

We start by defining the reasoning task. We then discuss example generation methods for three scenarios: a single rule as input, multiple (possibly conflicting) rules that require reasoning for the same conclusion, and multiple rules that require a sequence (chain) of reasoning steps. An example of a generated datapoint with a single rule is shown in Section 4.4.3. More examples in the appendix can be found for union of rules (Section B.6) and chained rules (Section B.7).

4.4.1 Reasoning Task

Each example is a triple *(context, hypothesis, confidence)*. *Context* is a combination of rule(s) and generated facts, such as "*If the first person lives together with the second person, then the first person is the spouse of the second person.*" and "*Anne lives with Mike.*" *Hypothesis* is the statement to be assessed based on the context, e.g., "*Laure is the spouse of Mike.*" *Confidence* is the probability that the hypothesis is valid given by the reasoner, e.g., 0.7. As we generate the examples, we know the confidence for each hypothesis.

4.4.2 Single-Rule Dataset Generation

Given a rule, we generate examples of different hypotheses to expose the model to various contexts. Each example contains the context c and a hypothesis h with its probability of

being true as obtained for the (c, h) pair from the LP^{MLN} reasoner. The intuition is that the examples show the expected behavior of a formal reasoner for every combination of possible facts for a given rule. This process is not about teaching the model specific facts to recall later, but 'teaching' it reasoning patterns.

Unlike previous work (Clark et al., 2020), our rules allow for multiple variables. This introduces additional complexity as examples must show how to deal with the symmetry of the predicate. For example, *child(Alice,Bob)* and *child(Bob,Alice)* are not equivalent since *child* is not symmetric, while *spouse(Alice,Bob)* and *spouse(Bob,Alice)* are equivalent as *spouse* is symmetric. We assume that metadata about the symmetry and the types is available from the KG for the predicates in the rules.

Given as input (i) a rule r, (ii) a desired number n of examples, (iii) an integer m to indicate the maximum number of facts given as a context, and (iv) a pool of values for each type involved in r's predicate *pools*, Algorithm 4 outputs a dataset D of generated examples.

Algorithm 4: Generate Synthetic Data

Input: rule r ; // child(a,b)→parent(b,a)
n ; // # of examples
m ; // max # of facts
pools ; // pools of names
Output: Generated Dataset D
1 $D = \{\}$, $i = 1$; // initialize
2 **while** $i \leq ceiling(n/8)$ **do**
3 $F = GenFacts(r, m, pools)$; // *child(Eve,Bob),parent(Amy,Sam)*
4 $O = LPMLN(r, F)$; // reasoner output
5 $h_1 = f \in F$; // *child(Eve,Bob)*
6 $h_2 = Alter(f)$; // *negchild(Eve,Bob)*
7 $h_3 = r(F)$; // *parent(Bob,Eve)*
8 $h_4 = Alter(r(F))$; // *parent(Eve,Bob)*
9 $h_5 = \text{pos}.f_l \notin F$; // *child(Joe,Garry)*
10 $h_6 = \neg h_5$; // *negchild(Joe,Garry)*
11 $h_7 = f_r \notin O$; // parent(Alice,Joe)
12 $h_8 = \neg h_7$; // *negparent(Alice,Joe)*
13 $D.add(h_{1-8})$;
14 $i \leftarrow i + 1$;
15 **Function** `GenFacts`(r,m,*pools*)**:**
16 $F = GetRandomFacts(r, pools, m)$;
17 $F.add(GetRuleFacts(r, pools))$;
18 **return** F
19 **Function** `Alter`($p(s, o)$)**:**
20 **if** *p is symmetric* **then return** $\neg p(s, o)$;
21 **if** *random()>0.5* **then return** $\neg p(s, o)$;
22 **else return** $p(o, s)$;

We start at line 3 by generating facts, such as *child(Eve,Bob)*, using the function

GenFacts (lines 15–18), which takes as input a rule r, the maximum number of facts m to generate, and the *pools*. A random integer less than m sets the number of facts in the current context. The generated facts F have predicates from the body of r, their polarity (true or negated atom) is assigned randomly, and variables are instantiated with values sampled from the pool (line 16). Facts are created randomly, as we are not interested in teaching the model specific facts to recall later, but instead we want to 'teach' it how to reason with different combinations of rules and facts. We then ensure that the rule is triggered in every context, eventually adding more facts to F using the function *GetRuleFacts* in line 17. After obtaining F, we feed rule r along with facts F to the LP^{MLN} reasoner, and we obtain a set O containing all satisfied facts and rule conclusions (line 4).

We generate different hypotheses, where each one leads to an example in dataset D. For each context, we add an example with different facts with respect to the given rule according to three dimensions. A fact can be (*i*) for a predicate in the premise or in the conclusion of a rule, could be (*ii*) satisfied or unsatisfied given the rule, and could have (*iii*) positive or negative polarity. This makes eight different possibilities, thus leading to the generation of eight different hypotheses (one for each context).

The first hypothesis h_1 is obtained by sampling a fact from the set F (line 5). We then produce the counter hypothesis h_2 by altering the fact (line 6) using the function *Alter* (lines 19-22). Given a hypothesis $p(s, o)$ (line 19), we return its negated form if p is symmetric (line 20). Otherwise, if p is not symmetric, we produce a counter hypothesis either by negation (line 21), or by switching the subject and the object in the triple as the predicate is not symmetric (line 22). We rely on a dictionary to check whether a predicate is symmetric or not.

We then produce hypothesis h_3 (line 7), which is the outcome of triggering rule r with the facts added in line 17. The counter hypothesis h_4 is generated by altering h_3 (line 8). Moreover, we generate hypothesis h_5 by considering any unsatisfied positive fact outside F. Following a closed-world assumption (CWA), we assume that positive triples are false if they cannot be proven, meaning that their negation is true. We sample a fact f_l from the set of all possible positive facts that do not have the same predicate of the rule head (line 9). Thus, h_5 will never be in the output O of the reasoner, as it cannot be derived. We then produce h_6 by negating h_5 in line 10. We further derive h_7 by sampling a fact f_r that has the same predicate as that of the rule head, but does not belong to the output of the reasoner O (line 11). For a positive (negative) rule, such a fact is labeled as False (True). h_7 is then negated to get the counter hypothesis h_8 (line 12). All generated hypotheses are added to D (line 13), and the process repeats until we obtain n examples. Finally, we automatically convert the examples to natural language using predefined templates. A basic template for atom predicate p (type t_1, type t_2) is "If the 1^{st} t_1 is p of the 2^{nd} t_2." ("If the first person is spouse of ..."). For the single-rule scenario, we release a dataset for 161 rules with a total of 3.2M examples and a 80%:10%:10% split for training, validation, and testing.

4.4.3 Data Generation Example

We show an example of data generation for Algorithm 4. For simplicity, here we show an example of a hard rule, i.e., one whose confidence is implicitly set to one.[1] We begin by setting the values of the input parameters:

Algorithm 1 Input:
- $r =$ `child(A,C)` $\wedge$ `parent(C,B)` $\rightarrow$ `spouse(A,B)`
- $n = 8$
- $m = 5$
- $pools =$`{Alice,Bob,Carl,David,Eve}`

We set $n = 8$ to generate all the eight hypotheses. We start by generating a set of facts F (line 3), having predicates from the body of the rule with random polarity. We ensure that there are facts that trigger the rule. Their number should not exceed m. Here is an example of generated facts F:

Generated Facts F:
- f_1: `negparent(Eve,Carl)`
- f_2: `child(Eve,David)`
- f_3: `parent(Carl,Bob)`
- f_4: `child(Alice,Carl)`

Four facts are generated in total. Facts f_3 and f_4 trigger rule r. We then feed the rule r and facts F into the LP$^{\text{MLN}}$ reasoner (line 4). The output O is then:

LP$^{\text{MLN}}$ Reasoner Output O:
- o_1: `child(Eve,David)`
- o_2: `child(Alice,Carl)`
- o_3: `parent(Carl,Bob)`
- o_4: `spouse(Alice,Bob)`
- o_5: `negchild(Eve,Carl)`

We start generating the hypotheses:

Generated Hypotheses H:
- h_1: `child(Eve,David)`
- h_2: `child(David,Eve)`
- h_3: `spouse(Alice,Bob)`
- h_4: `negspouse(Alice,Bob)`
- h_5: `child(David,Carl)`
- h_6: `negchild(David,Carl)`
- h_7: `spouse(Bob,Eve)`
- h_8: `negspouse(Bob,Eve)`

Hypothesis h_1 is obtained by sampling from F (line 5), and thus it is a valid hypothesis. Then, the hypothesis h_2 is generated by altering h_1 with the function *Alter* (line 19-22).

[1] We show an example of a soft rule in subsection B.6 below.

In this example, since *child* is not symmetric, h_2 is produced using a switch of the subject and the object of h_1 to generate a false hypothesis (line 6).

Hypothesis h_3 is the outcome of rule r being triggered by facts f_3 and f_4 (line 7). In a similar fashion to h_2, we produce h_4 (line 8).

Hypothesis h_5 is sampled from the universe of all unsatisfied positive facts having a different predicate than that of the rule body (line 9), which makes it an invalid hypothesis, as it is not found in the O. Hypothesis h_6 is the negation of h_5, and, following CWA, it is a valid hypothesis (line 10).

Finally, hypothesis h_7 is sampled from the universe of unsatisfied rule-head atoms (line 11), and it is negated to produce hypothesis h_8.

Overall, we obtain eight different examples represented in symbolic knowledge, where each example contains the set of generated facts F, the rule r, and a single hypothesis h_i. The following is one example in symbolic knowledge:

Example #1 (Symbolic):

- *Rule* r = `child(A,C)` $\wedge$ `parent(C,B)` $\rightarrow$ `spouse(A,B)`
- *Facts* F :
  - f_1: `negparent(Eve,Carl)`
  - f_2: `child(Eve,David)`
  - f_3: `parent(Carl,Bob)`
  - f_4: `child(Alice,Carl)`
- *Hypothesis* h_3 : `spouse(Alice,Bob)`

We then convert each example to synthetic English using a set of pre-defined templates for the facts and for the rules. Here is the above Example #1, but now rewritten in synthetic English:

Example #1 (Synthetic English):

- *Rule* r = If the child of the first person is the third person, and the parent of the third person is the second person, then the first person is the spouse of the second person.
- *Facts* F :
  - f_1**:** The parent of Eve is not Carl.
  - f_2**:** The child of Eve is David.
  - f_3**:** The parent of Carl is Bob.
  - f_4**:** The child of Alice is Carl.
- *Hypothesis* h_3 : The spouse of Alice is Bob.

The *Context* is defined as the combined set of facts and rule(s). Both *Context* and *Hypothesis* are fed as an input to the model.

Example #1 (Model Input):

- *Context* : The parent of Eve is not Carl. The child of Eve is David. If the child of the first person is the third person, and the parent of the third person is the second person, then the first person is the spouse of the second person. The parent of Carl is Bob. The child of Alice is Carl.
- *Hypothesis* : The spouse of Alice is Bob.

4.4.4 Rules with Overlapping Conclusion

When multiple rules are in the context, there could be facts that trigger more than one rule for a given hypothesis. The triggered rules might be all of the same polarity (positive or negative), eventually accumulating their confidence, or could be a mix of positive and negative rules that oppose each other. While the data generation procedure in Section 4.4.2 can be extended to handle multiple rules, this raises an efficiency problem. Given a set of R rules, it would generate $8^{|R|}$ examples for each *(facts,rule)* pair in order to cover all rule combinations. This is very expensive, e.g., for five rules, it would generate $8^5 = 32,768$ examples for a single context.

Given this challenge, we follow a different approach. We first generate data for each rule individually using Algorithm 4. We then generate more examples only for combinations of two or more rules having *all their rule conclusions* as hypotheses. For every input context, we produce rule-conclusion hypotheses (positive and negative) while varying the rules being fired. Thus, we generate $2 * \sum_{x=2}^{|R|} \binom{|R|}{x}$ examples with at least two rules triggered. Adding the single-rule data, we generate $8 * |R| + 2 * \sum_{x=2}^{|R|} \binom{|R|}{x}$ for every *(facts,rules)* pair, which is considerably smaller than 8^r for $|R| \geq 2$, according to the binomial theorem. For example, for $|R|$=5, we generate 92 examples per context. For the overlapping rules scenario, we release a dataset for 5 rules with a total of 300K examples, and a 70%:10%:20% split for training, validation, and testing. We use 5 rules, as it is enough to measure different combinations of rules (31 rule combinations), and it is very unlikely to have more than 5 rules sharing the same target predicate (Appendix B.2).

4.4.5 Chaining of Rule Executions

For certain hypotheses, an answer may be obtained by executing rules in a sequence, i.e., one on the result of the other, or in a *chain*. To be able to evaluate a model in this scenario, we generate hypotheses that can be tested only by chaining a number of rules (an example is shown in Appendix B.7). Given a pool of rules over different relations and a depth D, we sample a chain of rules with length D. We then generate hypotheses that would require a depth varying between 0 and D. We generate a rule-conclusion hypothesis (h_3) and its alteration (h_4) for each depth $d \leq D$. A depth of 0 means that the hypothesis can be verified using the facts alone without triggering any rule. We also generate counter-hypotheses by altering the hypotheses at a given depth, and we further include hypotheses that are unsatisfied given the input. For the chaining rules scenario, we start with a pool of 64 soft rules, and we generate hypotheses that would need at

most five chained rules to verify them. The dataset for $d \leq 5$ contains a total of 70K examples, and a 70%:10%:20% split for training, validation, and testing.

4.5 Teaching PLMs to Reason

In this section, we explain how we teach a PLM to 'reason' with one or more soft rules. Note that uncertainty stems from the rule confidence. One approach to teach how to estimate the probability of a prediction is to treat each confidence value (or bucket of confidence values) as a class and to model the problem as a k-way classification instance (or regression), but this is intractable when multiple rules are considered. Instead, we keep the problem as a two-class one by altering how the information is propagated in the model to incorporate uncertainty from the rule confidence.
Let $D = \{(x_i, y_i)\}_{i=1}^{m}$ be our generated dataset, where x_i is one example of the form *(context, hypothesis, confidence)* and y_i is a label indicating whether the hypothesis is validated or not by the context (facts and rules in English), and m is the size of the training set. A classifier f is a function that maps the input to one of the labels in the label space. Let $h(x, y)$ be a classification loss function. The empirical risk of the classifier f is $R_h(f) = \mathbb{E}_D(h(x, y)) = -\frac{1}{m}\sum_{i=1}^{m} h(x_i, y_i)$.

We want to introduce uncertainty in our loss function, using the weights computed by the LP$^{\text{MLN}}$ solver as a proxy to represent the probability of predicting the hypothesis as being true. To do so, we apply a revised empirical risk:

$$R'_h(f) = \mathbb{E}_D(h(x, y)) = -\frac{1}{m}\sum_{i=1}^{m}(w(x_i) * h(x_i, 1) + (1 - w(x_i)) * h(x_i, 0)) \tag{4.5}$$

where $w(x_i)$ is the probability of x_i being True.

We now state that each example is considered as a combination both of a weighted positive example with a weight $w(x_i)$ provided by the LP$^{\text{MLN}}$ solver and a weighted negative example with a weight $1 - w(x_i)$. When trained to minimize this risk, the model learns to assign the weights to each output class, thus predicting the confidence for the true class when given the satisfied rule head as a hypothesis.

4.6 Experiments

We first describe the experimental setup (Section 4.6.1). We then evaluate the model on single (Section 4.6.2) and on multiple rules (Sections 4.6.3 and 4.6.4). We show that a PLM fine-tuned on soft rules, namely RuleBERT, makes accurate predictions for unseen rules (Section 4.6.5), and it is more consistent than existing models on three external datasets (Section 4.7). We report the values of the hyper-parameters, as well as the results for some ablation experiments in Appendices B.3 and B.4 respectively. The datasets for all experiments are summarized in Table 4.1.

Dataset	Total	Train	Dev	Test
Single Rule (Section 4.6.2)	20K	16K	2K	2K
Overlap (Section 4.6.3)	300K	210K	30K	60K
Chaining (Depth=5) (Section 4.6.4)	70K	56K	4.6K	9.4K
RuleBERT (Section 4.7)	3.2M	2.56M	.32M	.32M

Table 4.1: Datasets for the experiments and their splits.

Rule	Conf.	*RoBERTa-wBCE* Acc.	CA@k .10	CA@k .01	*RoBERTa* Acc.	CA@k .10	CA@k .01
birthYear(a,c) ∧ deathYear(b,d) ∧ >(c,d)→negspouse(a,b)	.990	.995	.993	.993	.970	.490	.486
birthYear(b,d) ∧ foundYear(a,c) ∧ <(c,d)→negfounder(a,b)	.990	.928	.927	.927	.908	.486	.456
spouse(c,a) ∧ parent(b,c) → negspouse(a,b)	.923	.974	.963	.747	.875	.491	.279
relative(a,c) ∧ spouse(b,c) ∧ child(b,a) → relative(a,b)	.860	.922	.844	.801	.866	.342	.146
parent(c,a) ∧ child(b,c) → spouse(a,b)	.825	.944	.828	.444	.842	.342	.146
publisher(c,b) ∧ subsequentWork(c,a) → publisher(a,b)	.721	.909	.834	.765	.905	.358	.219
successor(b,a) → negspouse(a,b)	.718	.972	.896	.693	.949	.369	.313
child(c,b) ∧ relative(c,a) → negchild(a,b)	.644	.935	.880	.693	.905	.310	.303
child(c,b) ∧ spouse(a,c) → negrelative(a,b)	.562	.920	.907	.608	.915	.255	.250
relation(a,b) → negchild(a,b)	.549	.904	.886	.737	.902	.371	.366
child(c,b) ∧ spouse(c,a) → child(a,b)	.492	.901	.827	.422	.658	.223	.107
knownFor(b,a) → founder(a,b)	.387	.882	.601	.477	.839	.372	.215
founder(c,b) ∧ publisher(c,a) → negfounder(a,b)	.246	.886	.795	.665	.802	.311	.297
publisher(a,c) ∧ parentCompany(b,c) → negpublisher(a,b)	.235	.812	.748	.643	.811	.313	.271
successor(c,a) ∧ spouse(c,d) ∧ successor(d,b)→spouse(a,b)	.221	.927	.738	.628	.761	.248	.215
relative(a,c) ∧ parent(c,b) → child(a,b)	.135	.841	.704	.552	.727	.227	.182

Table 4.2: Evaluation results for single-rule models.

4.6.1 Experimental Setup

Rules. We use a corpus of 161 soft rules mined from DBpedia. We chose a pool of distinct rules with varying number of variables, number of predicates, rule conclusions, and confidences.

Reasoner. We use the official implementation[2] of the LP$^{\text{MLN}}$ reasoner. We set the reasoner to compute the exact probabilities for the triples.

PLM. We use the HuggingFace pre-trained *RoBERTa*$_{LARGE}$ (Liu et al., 2020b) model as our base model, as it is trained on more data compared to *BERT* (Devlin et al., 2019), and is better at learning positional embeddings (Wang and Chen, 2020). We fine-tune the PLM[3] with the weighted binary cross-entropy (wBCE) loss from Section 4.5. More details can be found in Appendix B.3.

Evaluations Measures. For the examples in the test set, we use accuracy (Acc.) and F1 score (F1) for balanced and unbalanced settings, respectively. As these measures do not take into account the uncertainty of the prediction *probability*, we further introduce

[2] http://github.com/azreasoners/lpmln

[3] The prompt is *<s>context</s></s>hypothesis</s>*.

Confidence Accuracy@k (CA@k), which measures the proportion of examples whose absolute error between the predicted and the actual probabilities is less than a threshold k:

$$CA@k = \frac{\#\{x_i, |w_i - \hat{w}_i| < k\}}{\#\{x_i\}} \tag{4.6}$$

where x_i is the ith example of dataset, w_i is the actual confidence of the associated hypothesis given by the LP$^{\text{MLN}}$ reasoner, $\hat{w}_i$ is the predicted confidence by the model, and k is a chosen threshold.

The measure can be seen as the ordinary accuracy measure, but true positives and negatives are counted only if the condition is satisfied, where lower values for k indicate stricter evaluation.

4.6.2 Single Soft Rule

We fine-tune 16 models for 16 different positive and negative rules (one model per rule) using 16k training samples per rule. We compare the accuracy of each model (*i*) without teaching uncertainty using binary cross-entropy (RoBERTa), and (*ii*) with teaching soft rules using wBCE.

Results. Every row in Table 4.2 shows a rule with its confidence, followed by accuracy and CA@k (for $k = 0.1$ and $k = 0.01$) for both loss functions. We see that models fine-tuned using *RoBERTa-wBCE* perform better on CA@k. In terms of *accuracy*, both models perform well, with *RoBERTa-wBCE* performing better for all rules. Interestingly, the best performing rules are two rules that involve comparison of numerical values (birth years against death and founding years), which suggests that our method can handle comparison predicates.

Test	**Size**	**F1**	**CA@.15**	**CA@.1**	**CA@.05**
r_1	1.6k	.990	.987	.986	.954
r_2	1.6k	.999	.997	.996	.946
r_3	1.6k	.995	.994	.994	.992
r_4	1.6k	.990	.989	.988	.935
r_5	1.6k	1	.999	.998	.979
U=2	20k	.985	.997	.993	.968
U=3	20k	.925	1	.998	.949
U=4	10k	.956	1	1	.988
U=5	2k	1	1	1	.980

Table 4.3: Results for a model trained on five rules sharing the same predicate, and tested on multiple test sets.

4.6.3 Rules Overlapping on Conclusion

The dataset contains five soft rules with *spouse* or *negspouse* in the rule conclusion, and a confidence between 0.30 and 0.87 (shown in Figure 4.2). We train a model on the dataset and test it (*i*) on a test set for each of the five rules separately, (*ii*) on test sets with U triggered rules, where $U \in \{2, 3, 4, 5\}$.

(r_1, .87) child(a,c) $\wedge$ parent(c,b) $\rightarrow$ spouse(A,B)
(r_2, .64) child(a,b) $\rightarrow$ negspouse(a,b)
(r_3, .3) relative(a,b) $\rightarrow$ spouse(a,b)
(r_4, ,78) child(a,c) $\wedge$ child(b,c) $\rightarrow$ spouse(a,b)
(r_5, .67) predecessor(a,b) $\rightarrow$ negspouse(a,b)

Figure 4.2: The five overlapping soft rules.

Results. Table 4.3 shows that the model achieves high scores both on the single test sets (top five rows) and on the sets with interacting rules. The test sets with $U = 2$ and $U = 3$ are most challenging, as they contain $\binom{5}{2} = 10$ and $\binom{5}{3} = 10$ combinations of rules, respectively, while the one with $U = 5$ has only one possible rule combination. The high scores indicate that PLMs can actually learn the interaction between multiple soft rules.

4.6.4 Rule Chaining

Here, we assess models fine-tuned on various chaining depths. We construct six datasets for this scenario with increasing depths ($D = 0$, $D \leq 1$, $D \leq 2$, $D \leq 3$, $D \leq 4$, $D \leq 5$), i.e., dataset $D \leq x$ contains hypotheses that need at most x chained rules. We thus train six models (one per dataset), and we test them (*i*) on their own test dataset (Test), (*ii*) on the test set with $D \leq 5$ that contains all examples up to depth 5 (All), and (*iii*) on test sets with a chaining of depth x (Dep$x$).

Data	Mod0	Mod1	Mod2	Mod3	Mod4	Mod5
Test	.996	.926	.883	.852	.856	.831
All	.589	.743	.772	.811	.831	.831
Dep0	.993	.974	.973	.982	.978	.973
Dep1	.264	.860	.884	.887	.889	.889
Dep2	.396	.655	.730	.751	.750	.720
Dep3	.438	.581	.636	.684	.690	.656
Dep4	.538	.468	.547	.626	.666	.627
Dep5	.552	.356	.496	.703	.785	.744

Table 4.4: F1 scores for models trained on varying depths and tested on six datasets. The boxed area indicates models tested on unseen chaining depths.

Results. The results are shown in Table 4.4. We can see that the models achieve high F1 scores on the respective test sets for Depth 0. The red borderline indicates F1 scores

Predicates	Rule	*FT-PLM*	RuleBERT$_{20}$	*FT*-RuleBERT$_{20}$
Known	child(a,b) → parent(b,a)	.719	.869	.989
	relative(a,b) → negspouse(b,a)	.885	.885	.963
	child(a,b) ∧ child(b,c) → negchild(a,c)	.835	.888	.918
	parent(a,b) ∧ parent(a,c) → spouse(b,c)	.754	.757	.814
	parent(a,b) → negchild(a,b)	.923	.933	.963
Unknown	knownFor(b,a) → founder(a,b)	.817	.795	.971
	worksFor(b,a) → negfounder(a,b)	.951	.915	.952
	occupation(a, b) → negalmaMater(a, b)	.939	.917	.972
	author(c,b) ∧ series(a,c) → author(a,b)	.965	.937	.989
	city(a,b) → negstate(a,b)	.923	.912	.971

Table 4.5: Evaluation on unseen rules (accuracy). The first group contains rules with predicates seen by RuleBERT among the 20 rules used for fine-tuning, while the second group has rules with unseen predicates.

for models tested on chaining depths higher than the ones they have been trained on. We see that Mod3 and Mod4 do fairly well on Depth 5. However, there is a decrease for higher depths, possibly due to the need for more training examples in order to learn such depths.

Moreover, since we sample a chain of rules each time, it is likely that every model has been trained on certain chains of rules. This yields lower scores in the constant-depth test sets as the models are being tested on unseen rule chains.

Note that Mod0 shows a counter-intuitive increase in the F1 score for higher unseen depths. Chaining soft rules may lead to a low probability for the associated hypothesis, and thus eventually to a *False* label. However, Mod0 is not trained on chaining and sees a hypothesis that requires chaining as an unsatisfied fact, thus eventually labeling it as *False*, while in fact it is the chaining of the soft rules that is the cause for this label. This is never the case with hard rules, as the actual label there would be *True*.

4.6.5 Testing RuleBERT on Unseen Rules

We have seen that a PLM can be successfully fine-tuned with rules. We now study the performance on the PLM after it has been fine-tuned on 161 (single) rules. We call this fine-tuned model RuleBERT.

We first evaluate RuleBERT on unseen rules. We fine-tune it with only twenty randomly selected rules (shown in Figure 4.3) and call it RuleBERT$_{20}$. We then select ten new rules divided into two groups: (*i*) five rules containing predicates that were used in the rules for fine-tuning RuleBERT$_{20}$, and (*ii*) five rules that share no predicates with the fine-tuning rules. For each rule in the test sets, we run a model fine-tuned (with 4k examples) only for that rule (FT-PLM), the model fine-tuned on the twenty original rules (RuleBERT$_{20}$), and the same model fine-tuned again for the rule at hand (FT-RuleBERT$_{20}$).

Results. Table 4.5 shows that RuleBERT$_{20}$ outperforms the fine-tuned model (FT-PLM) on the first group. Even though fine-tuned on 20 rules, it learned enough about

child(a,b) → negparent(a,b)
child(a,b) → nespouse(a,b)
child(a,b) → negchild(b,a)
child(a,b) → negrelation(b,a)
parent(a,b) → negparent(b,a)
parent(a,b) → nespouse(a,b)
spouse(a,b) → relative(b,a)
successor(a,b) → predecessor(b,a)
predecessor(a,b) → negsuccessor(a,b)
successor(a,b) → negspouse(a,b)
predecessor(a,b) → negspouse(a,b)
child(a,c) ∧ parent(c,b) → spouse(a,b)
child(b,a) ∧ child(c,a) → spouse(b,c)
parent(a,b) ∧ parent(b,c) → negparent(a,c)
parent(a,b) ∧ child(c,a) → spouse(b,c)
spouse(a,b) ∧ parent(c,a) → negspouse(b,c)
spouse(a,b) ∧ child(a,c) → negspouse(b,c)
successor(a,c) ∧ successor(b,c) → negspouse(a,b)
publisher(c,b) ∧ subsequentwork(c,a) → publisher(a,b)
publisher(c,b) ∧ previouswork(c,a) → publisher(a,b)

Figure 4.3: The 20 random rules used for RULEBERT$_{20}$.

(*i*) symmetric/transitive predicates and (*ii*) rule confidence to predict correctly, even better than rule-specific models.

For the second rule group, the accuracy of RULEBERT$_{20}$ is high, but FT-PLM performs better. Applying the same fine-tuning on RULEBERT$_{20}$ yields the best results in all scenarios.

4.7 RULEBERT on External Datasets

As our fine-tuning propagates information in the layers of the encoder, we hypothesize that RULEBERT effectively "learns" logical properties of the concepts represented in the rules, such as negation and symmetry, and thus it could perform better on tasks testing such properties of PLMs. To study the negation of predicates, we use the *Negated LAMA datasets*, which test how PLMs distinguish a cloze question and its negation (Kassner and Schütze, 2020). In most cases, PLMs make the same prediction both for a positive statement ("*Relativity was developed by Einstein.*") and for its negation ("*Relativity was not developed by Einstein.*"). To test the symmetry relationship between predicates, we use the SRL test in *CheckList* (Ribeiro et al., 2020), which focuses on behavioral testing of NLP models; we use its test set for the duplicate-question detection task (QQP) (Wang et al., 2018a). Finally, we test deductive reasoning on the *bAbI* dataset and its Task #15 (Weston et al., 2016).

		Facts	RoBERTa		RuleBERT	
			ρ	%	ρ	%
GR	birthplace	2,404	90.99	18.51	71.72	4.20
	birthdate	1,565	82.87	1.40	63.55	0.13
	deathplace	649	86.44	0.31	71.13	0.00
T-REx	1-1	973	78.95	61.38	51.21	32.96
	N-1	20,006	87.56	43.80	67.63	11.48
	N-M	13,096	89.39	50.78	72.59	28.90
ConceptNet	—	2,996	42.61	9.00	37.43	4.83
SQ	—	286	89.71	44.76	75.05	26.32

Table 4.6: Negated LAMA: Mean Spearman rank correlation (ρ) and mean percentage of overlap in the first ranked predictions (%) for original vs. negated queries.

4.7.1 Negated LAMA Experiments

For Negated LAMA, we do not fine-tune RuleBERT for the task; instead, we replace its original classification layer by an MLM head with weights identical to those of RoBERTa (not fine-tuned). Note that this configuration is biased in favor of RoBERTa, as the parameters of the MLM head and of the RoBERTa encoder have been trained in conjunction and thus more optimal network weights have been found for this combination, which is not the case for our RuleBERT.

Results. Yet, even in this arguably unfair setting, RuleBERT outperforms RoBERTa on all datasets of Negated LAMA, as shown in Table 4.6. We can see that RuleBERT performs better on both evaluation measures used in (Kassner and Schütze, 2020). It achieves a lower mean Spearman rank correlation (ρ) and a much smaller percentage of positive and negated answers overlap (%). The correlation measure helps capture cases where negation has a small effect from those where it has a larger effect.

4.7.2 CheckList QQP Experiments

The CheckList tests (Ribeiro et al., 2020) have shown that PLMs fail in many basic cases. We hypothesize that RuleBERT can perform better on tasks and examples that deal with symmetric and asymmetric predicates, if such predicates have been shown to it during pre-fine-tuning. We experiment with the QQP dataset, which asks to detect whether two questions are duplicates. We identify a few rules that can teach a model about symmetric predicates, and we pre-fine-tune RuleBERT on them; then, we fine-tune it on the QQP dataset.

Results. Table 4.7 shows the results on the challenging CheckList QQP test set: we can see that RuleBERT achieves accuracy of 0.422 after one epoch, while RoBERTa is at 0.0. However, after three epochs RuleBERT is also at 0.0,[4] i.e., it started to unlearn what it had learned at pre-fine-tuning (Kirkpatrick et al., 2017; Kemker et al., 2018; Biesialska

[4]On the much easier QQP test set, RuleBERT achieved 0.89 accuracy after one epoch, and 0.91 after three epochs.

	Fine-Tuned	RoBERTa	RuleBERT
bAbI	1 epoch	.401	.477
	2 epochs	.676	**.863**
	3 epochs	.827	.825
Neg. LAMA	-	.684	**.852**
CheckList QQP	1 epoch	.000	**.422**
	3 epochs	.000	.000

Table 4.7: Evaluation on external datasets (accuracy).

et al., 2020). Learning a new task often leads to such catastrophic forgetting (Ke et al., 2021b). While there are ways to alleviate this (Ke et al., 2021b), this is beyond the scope of this work.

4.7.3 bAbI Task #15 Experiments

Finally, we experiment with task #15 of the bAbI dataset, where the goal is to assess whether a model can perform deductive reasoning. However, as mentioned in the original bAbI paper (Weston et al., 2016), it is not only desirable to perform well on the task, but also to use the fewest examples.

Thus, we use the smallest dataset consisting of about 2,000 data points. We hypothesize that, under the same conditions and hyper-parameters, RuleBERT should be able to generalize faster and to learn in fewer epochs. As PLMs produce varying scores when fine-tuned on small datasets, we repeat the experiment ten times, and we report the average scores (more details in Appendix B.5). We then compare to RoBERTa. Both models contain two classification layers to predict start and end spans of the input context.

Results. We can see in Table 4.7 that RuleBERT achieves accuracy of 0.863 in two epochs, while RoBERTa achieves 0.676. On the third epoch, RoBERTa catches up with accuracy of 0.827, while RuleBERT starts to overfit (goes down to 0.825), indicating that fewer epochs should be used.

4.8 Retrieving Explanations

To promote more the use of soft rules, we argue that rule weights can help guide users to the triggered rules, if any. In a single rule setting, this is trivial, as if the output probability matches the rule confidence, the user can be informed that the associated rule was triggered. In a multi-rule setting, the output probabilities can also guide the users into which rules were triggered. Assuming that rules r_1 and r_2 of Figure 4.3 were the only rules triggered, we would expect an output probability of 0.71 for a non-negative hypothesis. Thus, one could generate a mapping between triggered rules and output probabilities, such as that in Table 4.8, to provide rule explanations. Such a table can be generated during data generation and cached for later use to obtain rule explanations with a best-case time complexity of $O(1)$, as opposed to high-complexity brute force methods

that remove one sentence at a time (Clark et al., 2020). For rule combinations with relatively similar confidences ($(r_1, r_2)\&(r_1, r_3)$), the brute-force method could be applied but with only two rule combinations, i.e., we prune the set of rule combinations using the expected output confidence. Future work could include utilizing encoder-decoder models to generate explanations (Tafjord et al., 2021).

Rules	**Expected**
$\mathbf{r}_1$	0.87
$\mathbf{r}_2$	0.64
$\mathbf{r}_3$	0.30
$(\mathbf{r}_1, \mathbf{r}_2)$	0.71
$(\mathbf{r}_1, \mathbf{r}_3)$	0.74
$(\mathbf{r}_2, \mathbf{r}_3)$	0.13
$(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$	0.51

Table 4.8: Rule combinations and their expected output confidence.

4.9 Conclusion

We studied whether PLMs could reason with soft rules over natural language. We experimented with one flavor of probabilistic answer set programming (LP^{MLN}), but other semantics can be also used with the proposed methodology. We further explored the inference capabilities of transformer-based PLMs, focusing on positive and negative textual entailment. Augmenting PLMs with logical rules can help alleviate some of their logical shortcoming, which is a crucial aspect during claim verification in fact-checking. However, our work dismisses the challenge of retrieving the facts themselves. One might consider using PLMs for fact retrieval as they could act as knowledge bases (Petroni et al., 2019). Alas, querying PLMs might not be always straightforward, especially when their output does not match what a user was expecting (Figure 1.1(D)). In the following chapter, we introduce a new kind of embedding that steers the PLM output to match a user's intent.

5 Type Embeddings: Encoding Type Information for Pre-trained Language Models

One key component of fact-checking is the ability to retrieve facts efficiently. While facts could be stored in textual corpora or databases, PLMs can themselves act as means to store relational knowledge (Petroni et al., 2019). However, one challenge is that they are schema-less, meaning they do not explicitly define types of their entities. Figure 1.1(D) shows a PLM returning death places for a given prompt, where a user was expecting death years. Given that PLMs encode semantic types, such as 'European City' or 'Woman', it is beneficial to try and use such concepts to steer the model output. In this work, we introduce Type Embeddings (TEs), input embeddings that promote desired types in a PLM. Our proposal is to define a type by a small set of word examples. We empirically study the ability of TEs both in representing types and in steering masking predictions without changes to the prompt text in BERT. Finally, using the LAMA datasets, we show how TEs highly improve the precision in extracting facts from PLMs. Such improvements render PLMs as better fact-retrievers and thus can enhance overall fact-checking systems.

5.1 Introduction

PLMs based on transformers (Vaswani et al., 2017) have achieved state-of-the-art results in several downstream NLP tasks (Devlin et al., 2019; Liu et al., 2020b). Being trained in a self-supervised fashion, such models convey, to a certain extent, linguistic (Puccetti et al., 2021; Lin et al., 2019) and factual knowledge (Rogers et al., 2020; Meng et al., 2022). Being able to faithfully extract the desired knowledge is a crucial aspect that has sparked lots of interest (Petroni et al., 2019; Bouraoui et al., 2020).

However, querying the PLM for information is not always reliable and requires more than a manually-written prompt as an input (Petroni et al., 2020). This is opposed to a standard KG, where users formulate a structured SPARQL query specifying exactly what to expect at the output. For example, the query "`SELECT ?x WHERE wd:Q76 wdt:P26 ?x`" returns the spouse of Barack Obama, "Michelle Obama". In the PLM setting, the SPARQL query could be replaced by a natural-language prompt, such as "The spouse of

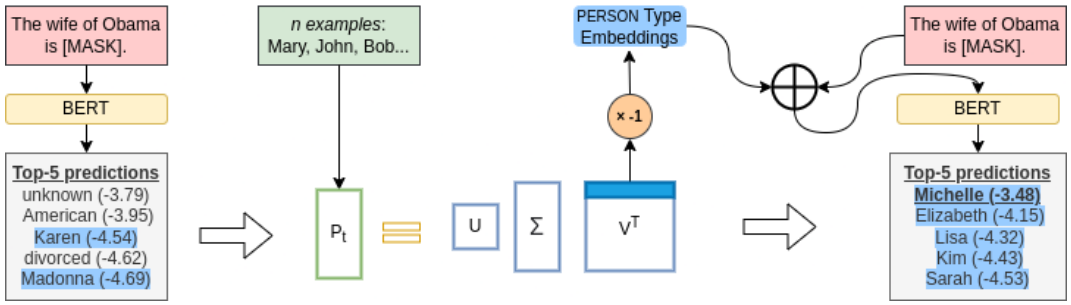

Figure 5.1: Top-5 predictions of BERT (with log probabilities) for a given prompt (left) and the changes when adding type information (right). Tokens following the desired type are colored. Correct answer is underlined.

Barack Obama is [MASK]". While the predictions of the prompt are reasonable (left-hand side of Figure 5.1), they do not reflect the requirement of getting instances of a specific type (names of people) in the output. In fact, analyzing BERT's top-1 prediction on prompts where the desired output type is a MUSICAL INSTRUMENT (e.g., "Philip Glass plays [MASK]"), more than half of the predictions follow different types such as SPORT ("plays football") and CHARACTER ("plays Hamlet"), instead of the expected "plays piano". Indeed, differently from the KG with typed entities and predicates, the output type information is dismissed from the input prompt, thus bringing no guarantee about the expected type.

While several works try to remedy this by engineering prompts to satisfy a desired type (Jiang et al., 2020; Shin et al., 2020; Zhong et al., 2021), or relying on external sources to enrich the prompt (Petroni et al., 2020), these approaches do not fully exploit the latent concepts encoded in the PLM (Dalvi et al., 2022; Mamou et al., 2020), and are susceptible to exploiting regularities of their training datasets such as predicting the majority class label (Zhong et al., 2021). To fill up this gap, we introduce the notion of *Type Embeddings* (TEs). Similar to how positional embeddings in a PLM encode information about the position of a token in an input (Wang and Chen, 2020), TEs encode the expected type information of the output. The definition of a TE requires neither supervised training nor external resources as it simply uses the existing PLM token embeddings, e.g., people names, to obtain type information, e.g., for PERSON. The TE can be then naturally injected into the input embedding layer of a PLM to embody the expected type in the output (right-hand side of Figure 5.1). Driving the model towards the expected type can help in numerous applications exploiting PLMs, such as data cleaning (Narayan et al., 2022), rule induction (Cui and Chen, 2021), and fact-checking (Lee et al., 2020).

Our contributions can be summarized as follows:

- We introduce TYPE EMBEDDINGS (TEs), which, similar to positional embeddings, can be added to the input of PLMs and effectively encode type information. We show how to compute these embeddings using only labeled tokens that adhere

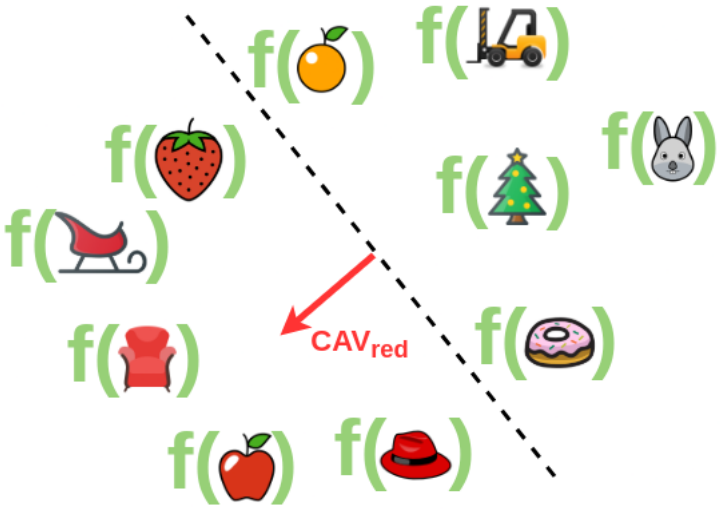

Figure 5.2: Given positive and negative examples of a concept (red), CAV_{red} is the normal to the hyperplane separating the examples. The function f is a model's activations at a certain layer.

to the specific type; the main idea is to remove the first singular vector (with a multiplier) of the token embedding matrix (Section 5.3).

- We propose methods to analyze type embeddings and evaluate their effectiveness by (i) measuring their semantic similarity to instances of the type, (ii) assessing the sensitivity of tokens to a given type, and (iii) analyzing layer-wise type classification (Section 5.4).
- We inject type embeddings into PLMs and show increase in performance for a factual probing dataset (LAMA) and alleviation of "type bias" for a prompt by steering the output type with TEs (Section 5.5).

We conclude the chapter by discussing future directions, including the extension of our approach from types to more generic concepts (Section 5.6). The data and the code are available at https://github.com/MhmdSaiid/TypeEmbedding.

5.2 Related Work

PLMs have been largely studied in the last years, with most analysis focusing on the attention mechanism (Voita et al., 2019; Vig and Belinkov, 2019; Kobayashi et al., 2020) and on the role of embeddings (Rogers et al., 2020; Li et al., 2021a; Clark et al., 2019b). However, none of those efforts study the notion of types that we introduce with our work. One exception is the recent studies of how *concepts* are encoded in PLMs. One line of work analyzes BERT by clustering contextual representations across layers, followed

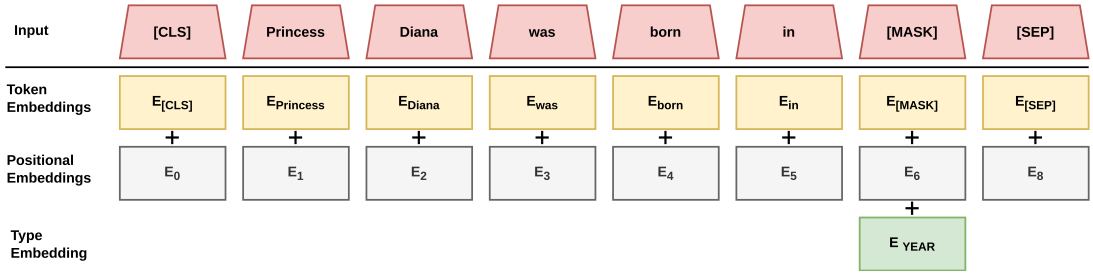

Figure 5.3: Input representation for a PLM (BERT). The YEAR type embedding (green box) is added to the [MASK] token, similarly to token and positional embeddings at the input.

by a manual annotation to label clusters with meaningful concepts (Dalvi et al., 2022). Another work starts from treating the feedforward network of a transformer as a key-value memory and studies how certain vectors encode concepts in the vocabulary space (Geva et al., 2022). Our effort is different in two ways. First, we do not require the labeling of artifacts from the PLM, but rather we rely on user-specified tokens to model their common type. Second, we focus on type, which is one semantic concept, leaving the others, such as syntactic, morphological, and lexical to future work (Section 5.6).

In the context of image classification, our approach is related to the interpretation of a neural net's internal state in terms of a concept defined through a vector (Kim et al., 2018; Schrouff et al., 2021). This is denoted as Concept Activation Vector (CAV) and is derived from example images by finding the normal to the hyperplane. CAVs separate examples with and without the target concept in a model's activation at a certain layer. Figure 5.2 shows the CAV for the concept 'red' in a certain model's layer. By Testing with a CAV (TCAV), one can identify the importance of the color 'red' in fire-engine images for a neural network. We use CAVs on textual input, rather than on images, to measure how sensitive the model is to a type after adding its TE (Section 5.4.3). However, while CAV is a sensitivity measurement tool, TEs steer the target type in the model's output.

A work sharing the same spirit as ours uses a vector to steer output in a PLM for style transfer between sentences (Subramani et al., 2022). However, we use tokens to obtain our vector and not sentences. We require 10 labeled tokens as opposed to 100 labeled samples for style transfer. The paper states that they 'cannot steer GPT-2 at either the embedding ... locations.', but our method could do that from the embedding layer.

Our work introduces a new kind of type embeddings to improve the input to the PLM in the target task, in analogy to what is done with positional embeddings (Wang and Chen, 2020; Wang et al., 2021a). To show the benefit of such a solution, we focus on the LAMA benchmark (Petroni et al., 2019), which is composed of a series of cloze statements used to query the PLM for a masked token, thus assessing a PLM's factual knowledge. To enhance PLMs' performance for such task, considerable work went into improving prompts by mining or paraphrasing new prompts (Jiang et al., 2020), by adding trigger

tokens (Shin et al., 2020), by finding vectors for prompts in the embedding space without restriction to the space of a PLM's vocabulary (Zhong et al., 2021), or by combining multiple prompts (Qin and Eisner, 2021). Our approach does not aim at refining the prompt, as we simply add the type embedding to the input. It is therefore also different from approaches that pre-train an adapter to enhance PLMs' factual knowledge (Wang et al., 2021c) or rely on information retrieval to provide additional context for the prompt (Petroni et al., 2020). Finally, we are steering the output while not changing the underlying model, for example by triggering the neurons responsible for a prediction (Dai et al., 2022) or producing an alternative model with edited facts (De Cao et al., 2021; Mitchell et al., 2022).

5.3 Type Embedding

In this section, we propose how to compute TEs from PLM token embeddings (Section 5.3.1), and how to use them (Section 5.3.2). Following the work on latent concepts in BERT (Dalvi et al., 2022), we focus on such model and report results on other PLMs in Appendix C.3.

5.3.1 Obtaining the TE

Given a type t, let the matrix $P_t \in \mathbb{R}^{n \times d}$ hold the token embeddings for n different tokens, where d is the dimension of the token embeddings. The n tokens are instances of a specific type t. We call these tokens *positively typed tokens*.

For our analysis of P_t, we apply *Singular Value Decomposition* (SVD). The SVD of an $m \times n$ matrix M factorizes it into $M = U\Sigma V^T$, where U is an $m \times m$ unitary matrix, Σ is an $m \times n$ diagonal matrix, and V is an $n \times n$ unitary matrix. We call the column vectors of U and V *singular vectors*. The diagonal values in Σ are called *singular values*. Assuming that M is a matrix where each row contains features of a data point, then the first singular vector of V, corresponding to the highest singular value, corresponds to the direction with maximum variance for the covariance matrix. In other words, it is the vector that contains the "common-part" of all data points.

The SVD of the matrix is $P_t = U\Sigma V^T$. The first column of the matrix V, $v^{(1)}$, is the first singular vector, which encodes information common between all n tokens. We hypothesize that this vector, unlike other singular vectors, contains non-type related information and needs to be removed from the input to promote type information encoded in the other singular vectors (more details in Section 5.4). A similar observation has been made for multilingual representations (Roy et al., 2020), where removing r singular vectors leaves semantic-related information in the input representations (Yang et al., 2021). Thus, the embedding to be added to promote type t is $E_t = -\lambda v^{(1)}$, where λ is a multiplier that is tuned on a hold-out dataset.

In practice, a type embedding is derived from a small set of tokens that are instances of the same type. Those can be provided by users, or obtained from existing typed resources such as KGs. In the rest of the chapter, the TEs are computed based on

weighted sampling from KG entities. We query the KG (DBpedia (Auer et al., 2007)) for tokens adhering to a specific type, keeping only those in the PLM's vocabulary, and use their node degree as the weight.

5.3.2 Using the TE

Assuming that a user has obtained the TE for the expected output type, the TE is simply added to the [MASK] input embedding, in analogy to token and positional embeddings. Figure 5.3 shows an example for a prediction where we enforce a YEAR type.

Depending on the task at hand, the TE can be added to one or more tokens. We found it more effective to add it only to the [MASK] token for MLM tasks, while for text generation it is more effective to add the TE to all tokens in the prompt. While our focus in this paper will be on MLM, we perform some preliminary experiment for text generation in Section 5.6.

Type Emb.	Predictions
CITY	*Kazan*(.69), *Baku*(.67), *Cologne*(.67), Düsseldorf(.63), *Toulouse*(.62), Strasbourg(.62), *Bonn*(.61)
YEAR	*1823*(.85), 1834(.83), 1819(.82), *1755*(.82), 1825(.82), *1835*(.82), 1805(.82)
OCCUPATION	*geologist*(.76), *biologist*(.73), *theologian*(.72), *screenwriter*(.7), botanist(.69), linguist(.68), novelist(.67)

Table 5.1: Most similar token embeddings to a given Type Embedding with cosine similarity score in parentheses. Tokens in italic were used to compute the TE.

5.4 Analysis of TEs

Having obtained a TE, we propose a series of analysis methods to assess its validity. We use the TE as a simple type retriever (Section 5.4.1), study the distribution of singular vectors (Section 5.4.2), analyze the effect of the TE w.r.t. the output and quantify the model's sensitivity w.r.t. typed tokens (Section 5.4.3), perform layer-wise classification to identify the desired type (Section 5.4.4), and measure TCAV of a model equipped with a TE (Section 5.4.5).

5.4.1 Similarity

As the TE is computed from token embeddings, the vector lives in the subspace formed by these embeddings. Therefore, we can use the TE to sort token embeddings (through cosine similarity) as a qualitative confirmation that the TE reflects the desired type. Table 5.1 shows examples of TEs for three types (cities, years, and occupations) and the most similar token embeddings of BERT. This suggests that TEs could act as a standalone type retriever, to sort tokens according to type and analyze any biases in the tokens from which the TE is computed. Applying the method on the first singular vector (i.e., $-E_t$), we observe that the top retrieved tokens ('.', 'and', 'the',...) relate to syntax, suggesting that the first singular vector encodes syntactic aspects, in agreement

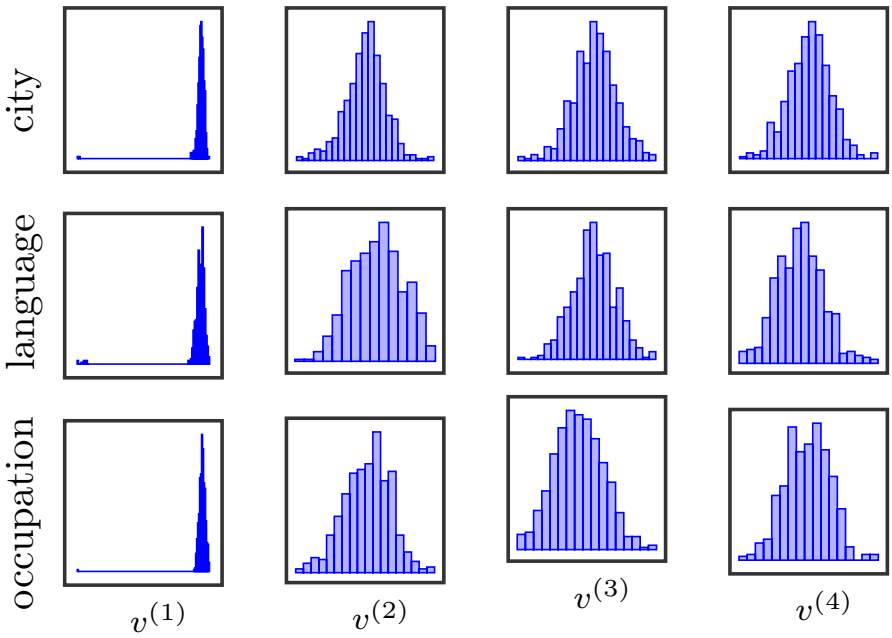

Figure 5.4: Distribution of the mean of the singular vectors across different types. We report the singular vectors with top-4 singular values. The distribution of the $v^{(1)}$ resembles less a Gaussian distribution (high kurtosis) as opposed to the others.

with other work in multilingual representations (Roy et al., 2020), showing that the first singular vectors encode non-semantic-related information (Yang et al., 2021).

5.4.2 Distribution of Singular Vectors

To understand the bias imposed by the first singular vector, we follow other work in analyzing distributions of singular vectors (Shin et al., 2018), where it is shown that distributions of singular vectors deviating from a Gaussian distribution contain bias.

From Figure 5.4, we see that the distribution of the singular vector $v^{(1)}$, corresponding to the largest singular value, clearly deviates from a Gaussian distribution, while others do not. This is indicated by the high kurtosis values for the first singular vectors. This suggests that this singular vector could represent a common bias that affects tokens (Shin et al., 2018). Note that since each singular vector is of dimension d, and to plot the histogram, we report the mean of the singular vector.

5.4.3 Effect of TE

We introduce two metrics for measuring TE's effectiveness.

Adversarial Accuracy. We expect that adding a TE to BERT causes the PLM to be more "type aware" in the associated task, i.e., adding the TE conveys type-related

Type	*AA*	*AS*
CITY	.853 (.0166)	.82 (.014)
LANGUAGE	1 (0)	.860 (.012)
OCCUPATION	1 (0)	.893 (.018)

Table 5.2: Mean and standard deviation (in parentheses) of *AA* and *AS* for different types ($\lambda = 1$).

tokens in the output. For example, in an MLM task, adding the TE should rank higher the tokens following the associated type. In a NLG task, adding a TE should convey more type-related tokens in the generated text. We focus on the former in this work, and leave the latter for future work.

To validate this hypothesis, we check if the score of a positively typed token in an MLM task for a model with the associated TE is greater than that of a standard BERT model. Formally, given a model $\mathcal{M}_t$, with an MLM head that has been equipped with a TE E_t promoting a specific type t, we denote by $P^{(x)}_{\mathcal{M}_t}$ the normalized output score of the token x with model $\mathcal{M}_t$ and prompt pr. To assess the effectiveness of the TE, we compute this normalized probability to that of an adversary, a BERT model without any equipped TE. We define the metric *adversarial accuracy* (AA) as:

$$AA = \frac{|\{x \in X_{t^+} | P^{(x)}_{\mathcal{M}_t} > P^{(x)}_{\mathcal{M}_\emptyset}\}|}{|X_{t^+}|} \tag{5.1}$$

where $\mathcal{M}_\emptyset$ is a model without any TE, and X_{t^+} is a set of tokens adhering to the type t. A higher value indicates that the TE is able to promote PLM tokens following type t.

Adversarial Sensitivity. We also expect that adding the TE should make tokens following the type more sensitive to the input TE. In other words, adding type information should have a larger effect on positively typed tokens, as opposed to without. To validate this hypothesis, we compare the sensitivity of a token w.r.t. the input in two models with and without a TE. If the former is greater than the latter, then the model is more sensitive to the typed token.

More formally, given a model $\mathcal{M}$, the output score of a token x is $P^{(x)}(X_{[MASK]})$[1]. With a first-order Taylor series expansion, we obtain $S^{(x)}_{\mathcal{M}} = P^{(x)}(X_{[MASK]}) - P^{(x)}(\mathbf{0}) \approx \frac{\partial P^{(x)}(X_{[MASK]})^T}{\partial X_{[MASK]}} X_{[MASK]}$, where $\mathbf{0}$ is the zero vector. The zero vector follows naturally from the neural-network-pruning literature (LeCun et al., 1989; Molchanov et al., 2017), where the relevance of a parameter is measured by how much removing it affects the output. Here, the zero vector does not allow for contextualising information in the attention mechanism.

$S^{(x)}_{\mathcal{M}}$ is reminiscent of metrics used in the neural network pruning literature (LeCun et al., 1989; Molchanov et al., 2017). However, the metric is applied w.r.t. a vector rather

[1]Other input tokens are omitted for brevity.

than to the usual case of scalar, and we do not take the absolute value of the metric as we focus on comparing sensitivities of models and not measuring an absolute effect, i.e., we want to be sure that adding the TE increases the output score.

Finally, to test a TE, we compare the sensitivity to that of a standard BERT model. Similarly, we define *adversarial sensitivity* as the number of positive typed tokens whose sensitivity increased after adding TE to the number of positive typed tokens in a set X_{t^+}. More formally:

$$AS = \frac{|\{x \in X_{t^+} | S^{(x)}_{\mathcal{M}_t} > S^{(x)}_{\mathcal{M}_\emptyset}\}|}{|X_{t^+}|} \tag{5.2}$$

For both measures, we report results over a random sample of 100 tokens, making sure that every one is an instance of type t and none of them has been used to derive the TE. We then compute the accuracy 10 times to get mean and standard deviation. To make sure that any change in the scores is due only to the TE, we set $pr = [MASK]$. This simple prompt neglects any contextual information that might affect PLM tokens, thus ensuring that any change is due to the TE.

Results for mean and standard deviation are reported in Table 5.2 for both AA and AS. For AA, TEs perform well in promoting tokens respecting a certain type. We observe a lower score for type CITY, which is likely due to (a) the large cardinality of the CITY type making it more difficult to model all required aspects of cities, and (b) coincidence of some city tokens with people names such as Morris, Salem, and Riley.

For AS, the TE has a small error margin. As we cannot expect token embeddings to capture all intricacies of a certain type, there are examples where the model fails the sensitivity test. Examples of failing tokens that did not show improvement in type sensitivity for CITY are Salvador and Blair, for LANGUAGE are Cherokee and Romani, and OCCUPATION are general and vicar.

5.4.4 Layer-wise Classification

As TEs are added at the input of the model, we postulate that adding TEs should help BERT identify types of input prompts more efficiently. For this, we train a layer-wise linear classifier on embeddings of input prompts, where positive instances are prompts belonging to a certain type t and negative instances are prompts of other types (examples in Table 5.3). For each type, we randomly sample 100 positive and negative instances from other LAMA datasets (negative instances are sampled randomly from the remaining types), and train a layer-wise linear classifier. We repeat each experiment 10 times and report mean accuracy on a test set of the same size. Prompts appearing in the train set do not appear again in the test set. Results in Figure 5.5 show that adding TE gives most layer classifiers an increase in F1-score. The highest increase is usually at a layer in the middle, in agreement with other work (Dalvi et al., 2022), possibly because this is where a type is formed (Geva et al., 2021b; Jawahar et al., 2019). The highest increase is for LANGUAGE, likely due to the smaller cardinality of the type compared to CITY and ORGANIZATION. We obtain from these the classifiers the CAVs needed for TCAV in the following section.

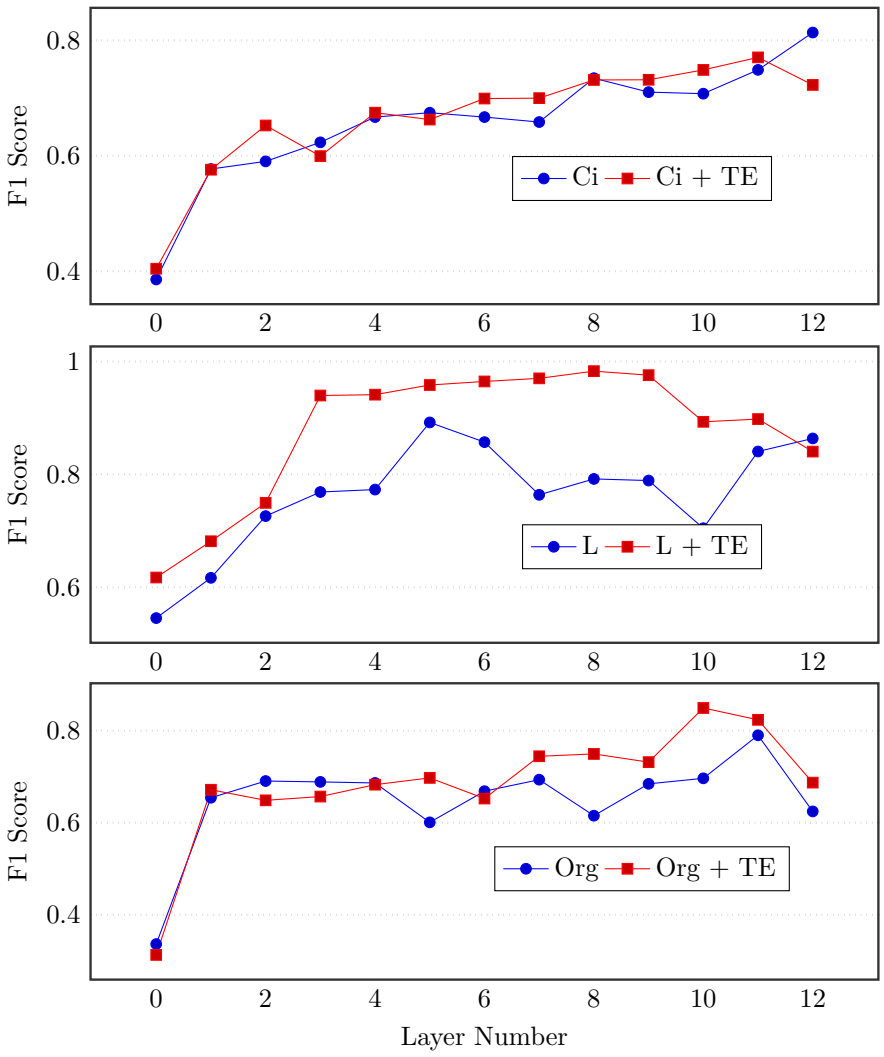

Figure 5.5: F1 scores of three classifiers trained and tested on layer-wise embeddings of CITY (Ci), LANGUAGE (L), and ORGANIZATION (Org) datasets.

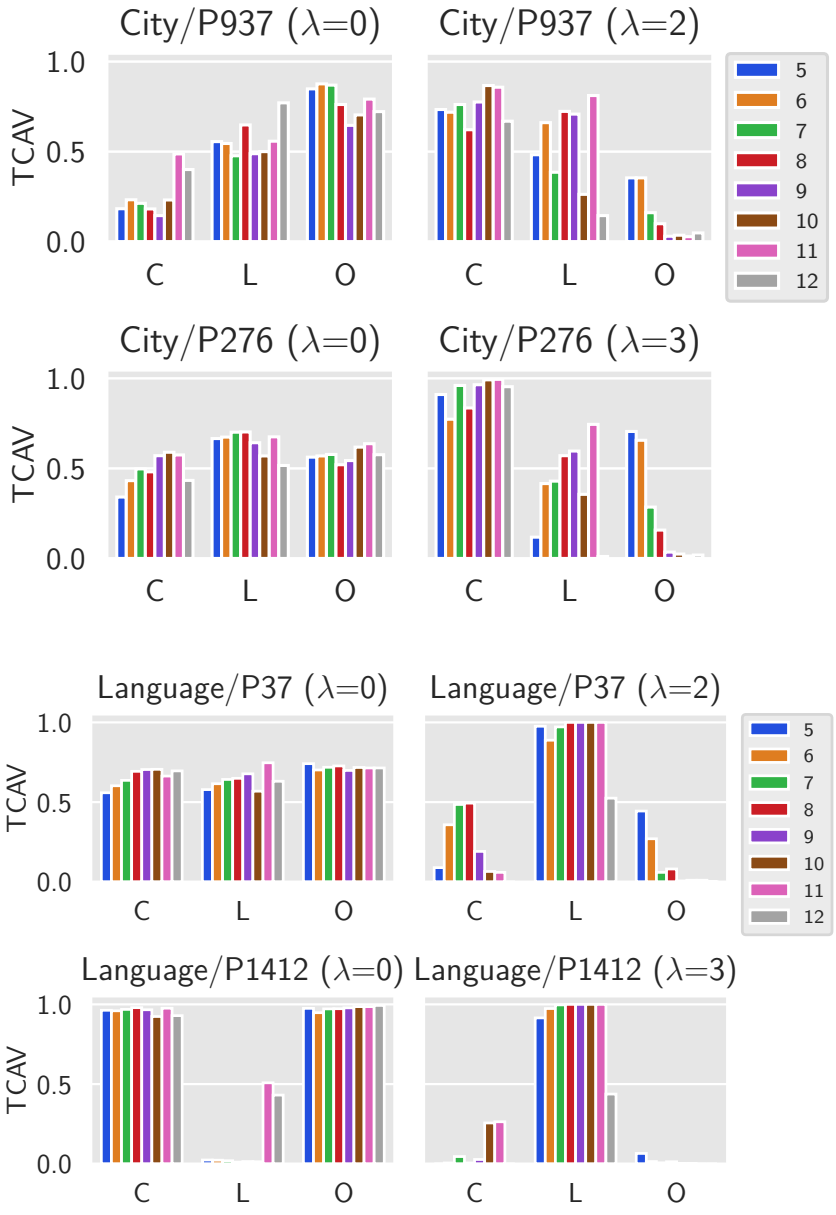

Figure 5.6: TCAV values for CITY (top) and LANGUAGE (bottom) datasets compared against the CITY (C), LANGUAGE (L), and ORGANIZATION (O) CAVs for layers 5-12 of BERT (left) and BERT+TE (right).

Dataset	Prompt Example
P27	Albert II of Belgium is [MASK] citizen .
P1376	Cardiff is the capital of [MASK] .
P17	Cairo American College is located in [MASK] .
P131	Saharsa district is located in [MASK] .
P20	Fredegund died in [MASK] .
P937	Xavier Zubiri used to work in [MASK] .

Table 5.3: Examples of LAMA datasets grouped by output types COUNTRY (top) and CITY (bottom).

5.4.5 TCAV Sensitivity

A *Concept Activation Vector* (CAV) for a concept is simply a vector in the direction of the values of that concept's set of examples (Kim et al., 2018). For example, given images showing the concept of the red color (positive samples), and other images that do not (negative samples), a linear classifier is trained on the activations at each layer when inputted by such images to separate the positive and negative samples (Figure 5.2). The normal to the hyperplane (CAV_{red} in Figure 5.2) separating both samples is the CAV. By using CAVs (with directional derivatives), one can measure the sensitivity of an input w.r.t to a concept by gauging the sensitivity of ML predictions to changes in inputs towards the direction of a concept. Thus, given a set of datapoints representing a certain concept, *Testing with CAVs* (TCAVs) provides means to compute the model's conceptual sensitivity across the input (Kim et al., 2018). As a final analysis measure, we posit that a model equipped with a TE should have higher TCAV values across layers. For this, we compute layer-wise TCAV using the CAVs in Section 5.4.4. Figure 5.6 shows the TCAV values for types CITY and LANGUAGE, comparing a vanilla BERT model ($\lambda = 0$) and one equipped with TE ($\lambda > 0$) for layers 5-12. As TCAV computes the model's conceptual sensitivity across a set of inputs, we observe that with the right TE, the importance of the type becomes more salient. This mean that the sensitivity of model predictions with respect to types, such as CITY at a certain layer, increases when inputting a prompt and a TE associated with that type.

5.5 Experiments

The LAMA benchmark (Petroni et al., 2019) contains cloze statements to test PLMs' factual knowledge. First, we apply TEs to PLMs and show increase in precision for most datasets (Section 5.5.1). We then enforce a change in the output with TEs (Section 5.5.2). Finally, we show the impact of the tokens that encode the TE (Section 5.5.3). We perform weighted sampling tokens from DBpedia where the weight represents the node degree of the token in the KG.

	P@1	P@10	P@50	P@100
B	.223	.509	.740	.845
BTo	.146	.327	.550	.640
PostTE	.248	.577	.819	.889
BTE (our method)	.291	.606	.838	.899

Table 5.4: Mean precision over all LAMA datasets compared to intrinsic baselines.

5.5.1 LAMA

We focus on the GRE and TREx datasets (Elsahar et al., 2018) as their prompts can be grouped into 17 output types from 38 datasets, with most examples covered by types CITY, LANGUAGE, and COUNTRY; examples for two types are in Table 5.3 (full list in Appendix C.1). We remove prompts whose expected output is not in BERT's vocabulary and prompts containing more than one [MASK] token. This gives an upper bound on BERT's performance on LAMA.

As stated in Section 5.3.1, the type embedding is computed with weighted sampling from KG entities (10, by default). To tune the value λ of a TE, we use a hold-out (dev) dataset of 5% for each dataset, and choose the λ value that maximizes precision. We report results mainly on a BERT BASE CASED model. Further experiments with BERT LARGE CASED and ROBERTA BASE show similar trends (Table C.3).

Intrinsic Evaluation. We compare BERT with TE (*BTE*) against standard BERT (*B*). As we assume that the user knows the desired output type, we also report for a baseline BERT + Token Type (*BTo*), which adds the expected type label (e.g., "the year") before the [MASK] token. We also report on a baseline *PostTE* which uses the TE at the output for re-ranking. The initial output score is added to the cosine similarity between the token embedding and the type embedding, controlled by a hyper-parameter to adjust the importance of the similarity score. We choose the range of the hyper-parameter to vary from 0 to 30 as in a similar work for natural language generation (Pascual et al., 2021). We also tested a second baseline where we add the tokens used to derive the TE before the [MASK] token, as a signal of the desired types (Shin et al., 2020), but the results are lower than BTo. Aggregated (macro) precision@k (P@k) results over all datasets are reported in Table 5.4 (full results in Appendix, Table C.4). On average, our proposal clearly improves the results. We see improvements across most of the types using TEs. However, we do observe reduction of precision in a few types, where the main reason being the greedy selection of a non-optimal value of λ. For type MANUFACTURER, setting $\lambda = 1$ (rather than $\lambda = 2$) improves the results. For type SPECIALIZATION, while desired outputs such as mathematics and physics do exist in the KG samples, other nodes in the KG, such as teenager, Greek, and Sir have greater node degree and thus got selected in the sample for obtaining the TE. For the GROUP data, the value of λ for the TE was 0, meaning that adding the TE would hurt performance. Analyzing the predictions, we believe this is due to the bias in the TE imposed by the KG as most samples are

	P@1	P@10	P@50	P@100
LPAQA	.288	.607	.791	.855
BTE	**.317**	**.650**	**.868**	**.920**
OptiPrompt	**.469**	**.790**	**.922**	**.956**
BTE	.356	.697	.876	.930

Table 5.5: Mean precision over all LAMA datasets compared to extrinsic baselines. We perform better than **LPAQA** which uses super-vised learning to optimize for prompt weights. The other supervised-learning approach, **OptiPrompt**, obtains higher precision as it searches for prompts in the embedding space. We report double entries for **BTE** since each baseline had different pre-processing functions, thus leading to different dataset sizes.

related to sport groups (such as FIFA, UEFA, and CONCACAF) thus producing a TE biased towards sports group which negatively impacts the predictions. We discuss other sampling methods in Section 5.5.3. Finally, the YEAR dataset shows lower performance. We believe this is due to BERT's inability to precisely capture numeracy (Wallace et al., 2019). For **PostTE**, our method, of using the TE at the input, produces better results, as using the TE at the output does not allow for the fusion between factual and type knowledge in the model. **PostTE** does push typed tokens to higher rankings (indicating also the effectiveness of TEs in modeling type), but adding TEs to the input is better in terms of performance. Plus adding TEs to the input is more universal:, as the output is usually controlled by the experiment type (binary classification, MLM, NLG,...), which might not always make it clear how to insert the TE, whereas the input is always fixed. One thing to note is that, with **PostTE**, out of the 38 different datasets used, 22/38 had an optimal value of λ to be zero. Meaning that for most datasets, it did not improve results, as opposed to our method which had only 5/38 datasets with optimal $\lambda = 0$.

Extrinsic Evaluation. We evaluate our model against two baselines. The first baseline, **LPAQA** (Jiang et al., 2020), uses mining-based methods on Wikipedia to identify possible prompts for a given relation. As not all mined prompts might be efficient for fact retrieval from PLMs, the authors propose to optimize for prompt weights to maximize scores on a training set. The second baseline, **OptiPrompt** (Zhong et al., 2021), follows other work in searching for prompts in the input embeddings space (Shin et al., 2020). However, rather then being constrained to discrete input tokens, **OptiPrompt** finds real-valued input vectors that maximize the likelihood of the gold label on the training set using a gradient-based searching algorithm. Results are found in Table 5.5. We observe that our approach does better with fewer prompts, as **LPAQA** requires at least 10X more prompts per example. For **OptiPrompt**, the supervised approach does produce better results, surpassing our method. However, the approach requires training data, which is not always available. The authors of the paper use only TREx relations as they can query the knowledge graph for more data, which is not the case for Google-RE

Prompt	TE	P@1	P@10	P@50	P@100
DoB	-	0	0	0	0
	E_{city}	.153	.404	.613	.701
	E_{city}^{optim}	.194	.444	.614	.719
PoB	-	.244	.533	.728	.808

Table 5.6: Precision in predicting PoB (place of birth) for DoB (date of birth) prompts by adding CITY TE ($\lambda = 5$). Results with TE are comparable to the PoB prompt.

datasets that we use. Also, the authors use 1000 data points for training, where one can not guarantee that 1000 data points are always available. For that, the authors had to rely on another knowledge graph to gather more samples. Our approach requires only 10 tokens per type belonging to the PLM's vocabulary. Finally, while training enhances performance, training data clearly encodes certain regularities that models could exploit, such as being prone to over-predicting the majority class label (**OptiPrompt** suffers from this (Zhong et al., 2021)), unlike our approach which keeps model parameters intact.

5.5.2 Switching Types in Prompts

LAMA authors provide manually written prompts that adhere to the desired type. For example, to get the PLACE OF BIRTH (PoB) of a person, they use the prompt "[X] was born in [Y].", while for the DATE OF BIRTH (DoB) of a person they use the prompt "[X] (born [Y])". These prompts follow from how sentences about date and place of birth are written in Wikipedia pages. In this experiment, we ponder whether TEs can enforce a different type given one of these two prompt structure. We use DoB prompts with the expected outcomes of PoB, where the goal is to steer the type of the output to a different type. For example, given "Barack (born [MASK])" (prompt for DoB), we set as expected output "Honolulu" (PoB answer). We remove examples for which the expected output is not in BERT vocabulary and are left with 1139 prompts. We then add the TE for CITY during inference. The results are shown in Table 5.6. As expected, without any TE, the precision score is zero as the output type is heavily influenced by the prompt. Adding E_{city} to the input steers the model to change type and it outputs cities. However, the scores are still less than those of PoB prompts. Since the prompt is biased towards a certain type, better results can be obtained by removing the projection of the year information onto the city TE. Our optimized TE is then $E_{city}^{optim} = E_{city} - \frac{E_{city}.E_{year}}{||E_{city}||_2||E_{year}||_2}E_{year}$, which indeed shows improved results in Table 5.6.

5.5.3 Token Sampling

We study the impact of how tokens for TEs are sampled by (i) changing the sampling method, and (ii) varying the number of tokens used.

	P@1	P@10	P@50	P@100
BTE	0.291	0.606	0.838	0.899
Top10	0.336	0.660	0.856	0.907
Bot10	0.235	0.534	0.764	0.846
Unif	0.250	0.563	0.798	0.884

Table 5.7: Mean over all datasets for every sampling method.

n	P@1	P@10	P@50	P@100
0	0.223	0.509	0.740	0.845
5	0.279	0.611	0.814	0.873
10	0.291	0.606	0.838	0.899
15	0.275	0.617	0.847	0.894
20	**0.298**	**0.644**	**0.859**	0.905
50	0.292	0.631	0.853	**0.906**

Table 5.8: Average of precision of the datasets while varying the number of samples n to compute the TE.

Sampling Methods. We evaluate forms of obtaining tokens alternative to weighted sampling: (i) weighted sampling with node degrees as weights (*BTE*), (ii) using the Top-10 tokens w.r.t. node degree (*Top10*), (iii) using the Bottom-10 tokens (*Bot10*), and (iv) sampling uniformly without relying on node degree (*Unif*). We repeat the experiment in Section 5.5.1 with every sampling strategy and show results in Table 5.7. More detailed results are found in Table C.5.

We observe that *Top10* and weighted sampling obtain comparable performance. While *Top10* gets better results for COUNTRY, ORGANIZATION, and GENRE, other types such as YEAR, SPECIALIZATION and MANUFACTURER show lower precision because of the bias coming from the most popular KG samples. For example, *Top10* samples only years in the 21^{st} century, specializations related to titles (duke, Sultan, and Sir rather than mathematics and physics), and it is biased towards car manufacturers (Fiat and Honda). Weighted sampling reduces such bias. For FOOTBALL POSITION, *Unif* does better as it has more variety in the sample with more tokens related to American football positions (quarter back and guard) rather than soccer positions only (goalkeeper and midfielder).

In some cases, the bias in the KG reflects the bias in the test data. For OCCUPATION, the TE using *Top10* does encode some bias as most tokens are related to artistic positions (musician, actor), but this improves results as the same bias occurs also among the expected outputs.

Varying Size of Samples. To study the effect of the number of tokens used in deriving the TE, we repeat the experiment in Section 5.5.1, while varying the number of tokens n. Results are reported in Table 5.8. We observe that results peak between 10 and 20 samples, but even a small number of samples significantly improves the results compared to the original BERT without TE (n=0). We note that as our method relies on SVD,

which is not a supervised-learning algorithm and there is no guarantee that the results would always improve with more data (tokens in our case). In fact, the impact of adding more tokens depends on what tokens are being added. Overall, according to Table 5.8, having more tokens enhances the results, but might introduce some bias when a large number is considered. However, even with 50 tokens, results are better than with 5.

5.6 Conclusion

Efficient fact retrieval leads to better fact-checking systems overall. One direction to achieve this is to encode types during PLM querying. This allows users to have more control on PLMs by defining their desired type and embedding it in the PLM. For this, we have introduced TEs as an additional input for PLMs to better encode type information, proposed methods to analyze TEs, and experimented with them on the LAMA dataset.

While initial results are promising, we identify two main directions of research.

More Precise Type Embeddings. Further analysis of the examples can lead to better type embeddings, which in turn lead to better fact-retrieval. One direction is to use also negative samples to compute the TE. This implies learning a vector that separates between the samples, as CAVs do. However, adding negative samples to the mix can bring more bias in the TE. This could be alleviated by performing some statistical hypothesis testing, as with CAVs (Kim et al., 2018).

Another way to improve the effectiveness of our proposal is to combine vectors. Assuming a taxonomy of the types, it would be possible to combine different TEs, for example by subtracting for the one at hand, say PERSON, all the ones that are not super or subtypes, such as CITY and YEAR, as we discussed for DoB in Table 5.6.

From Types to Concepts. While we focus on types and TEs, our approach can be extended to include more generic concepts, as long as tokens that relate to the concept are found in the PLM's vocabulary. This could help alleviate the stereotypical and toxic content found in NLP systems using PLMs (Ousidhoum et al., 2021). To test our idea, we report an example for the task of natural language generation, where we "de-toxify" text generated by an autoregressive language model. We use a distilled GPT-2 model (Radford et al., 2019) and the *RealToxicityPrompts* dataset that contains 100K sentence-level prompts derived from a corpus of English text (Gehman et al., 2020). We feed 10K samples to the model, thus producing the generated texts. We then measure the toxicity of such texts with the *Perspective API*[2]. We consider a text toxic if the toxicity probability returned by the API is > 0.5 and obtain 460 toxic prompts. We then compute a "toxicity concept embedding" using the method described in the chapter with 6 manually picked tokens that convey toxicity. To de-toxify the generated text, we set the multiplier λ to negative values. Instead of adding the embedding to the [MASK] token only, we found better results when adding it to all tokens in the prompt, for which we report results. We believe this is because adding the TE to all tokens helps to 'preserve'

[2] https://perspectiveapi.com/

	λ	**Toxicity (↓)** Toxicity pr.	**Fluency (↓)** Output ppl.	**Diversity (↑)** Di-1	Di-2	Di-3
Toxic	0	.687	4.727	.541	.455	.357
Prompt	-1	.389	6.340	.602	.476	.377
	-2	.356	17.564	.668	.509	.400
Non-toxic	0	.045	4.195	.801	.676	.528
Prompt	-1	.077	4.038	.782	.622	.484
	-2	.088	3.716	.840	.620	.475

Table 5.9: Results of detoxifying texts generated from a distilled GPT-2 model. λ indicates the value of the multiplier of the TE ($\lambda = 0$ for original PLM).

type information along the lengthy generation procedure, as opposed to MLM which decodes one token.

We also report on a sample of non-toxic prompts (size equal to that of toxic prompts) to show the effect of the concept embedding. In addition to toxicity, we measure also fluency (perplexity of generated continuations according to a larger PLM) and diversity (the mean number of distinct uni-, bi-, and trigrams, normalized by the length of text for each prompt), as it is common for evaluating text generation (Liu et al., 2021).

In the results in Table 5.9, we observe a huge reduction in the toxicity probability with $\lambda = -1$, higher more diversity but slightly less fluency for the toxic prompt. Setting $\lambda = -2$ decreases further the toxicity probability, but at the expense of less fluency. For the non-toxic prompts, the toxicity results are nearly the same, with minor differences for fluency and diversity. Considering that a "concept vector" steers the generation of the PLM without any form of fine-tuning, it is promising to study the use of "plug-and-play" concept vectors. A few examples are shown in Table C.2.

Despite enhancements in fact retrieval (this chapter) and claim verification (Chapter 4), relying solely on machines for fact-checking is ineffective, employing humans in inevitable. While humans are employed in Chapter 3, they are domain-specialized experts who are scarce and require formal training. As previous work has shown that a crowd of non-experts could be effective in misinformation identification (Allen et al., 2021a), it is promising to see how such a crowd would compare to professional experts in a large-scale uncontrolled environment. We study this in the following chapter.

6 Analyzing Large Scale Crowd-Sourcing in Twitter's BIRDWATCH

Fact-checking is one of the effective solutions in fighting online misinformation. However, traditional fact-checking is a process requiring scarce expert human resources, and thus does not scale well on social media because of the continuous flow of new content to be checked. Also, the methods of Chapters 3 and 4 focus on particular subsets of claims and require training data that, in a best-case scenario, would require labels of experts, thus rendering the fact-checking process costly and difficult to scale. Methods based on humans-in-the-loop have been proposed to tackle this challenge. Crowd-sourcing is when humans, typically with no professional expertise in fact-checking, are included in the process. It is a popular approach as the workers can scale with a smaller cost; but, while they have shown to be feasible, they have always been studied in controlled environments. In this chapter, we study the first large-scale effort of crowd-sourced fact-checking deployed in practice, started by Twitter with the BIRDWATCH program. Our analysis shows that crowd-sourcing may be an effective fact-checking strategy in some settings, even comparable to results obtained by human experts, but does not lead to consistent, actionable results in others. We processed 11.9k tweets verified by the BIRDWATCH program and report empirical evidence of i) differences in how the crowd and experts select content to be fact-checked, ii) how the crowd and the experts retrieve different resources to fact-check, and iii) the limits the crowd shows in fact-checking scalability and efficiency as compared to expert checkers.

6.1 Introduction

The spread of online misinformation carries risks for the democratic process and for a decrease in public trust towards authoritative sources of news (Starbird, 2019). Fact-checking is one of the prominent solutions in fighting online misinformation. However, traditional fact-checking is a process requiring scarce expert human resources, and thus does not scale well to social media because of the continuous flow of new content (Hassan et al., 2015). Automated methods and crowd-sourcing have been proposed to tackle this challenge (Thorne and Vlachos, 2018; Nakov et al., 2021a; Roitero et al., 2020a), as they

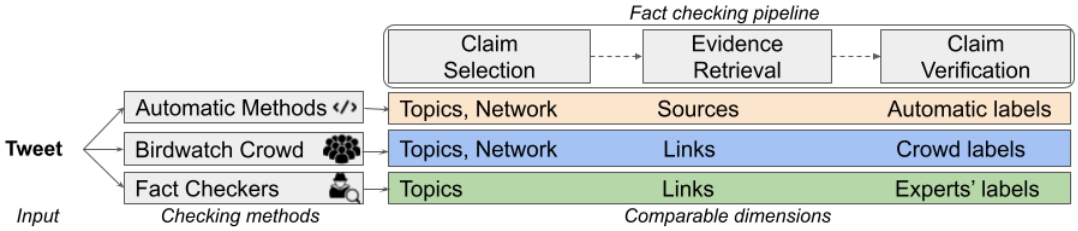

Figure 6.1: Given input tweets, the three alternative checking methods (automatic, crowd, professional checkers) are analyzed across their comparable dimensions according to a standard fact-checking pipeline.

can scale with a smaller cost, but have always been studied in controlled environments. Current approaches focus either on fully automated machine learning methods (Wei et al., 2019; Liu and Wu, 2018) or on hybrid human-machine approaches making use of crowd-sourcing to scale-up human annotation efforts (Roitero et al., 2020b).

The first large-scale effort of crowd-sourced fact-checking was piloted by Twitter with the BIRDWATCH program on the 23rd of January 2021 (Coleman, 2021b). BIRDWATCH adopts a community-driven approach for fact-checking by allowing selected Twitter users to identify fallacious information by (i) classifying tweets as misleading or not, accompanied by a written review, and by (ii) classifying reviews of other BIRDWATCH users as being helpful or not. In this setting, any user can create a *note* for a tweet (providing some metadata about the annotations) and other users can up/down *rate* such note. Multiple users can check the same content independently.

In this study, we perform an analysis of how crowd-sourced fact-checking works in practice when compared with human experts and automated fact-checking methods. To this end, we perform an analysis of the grass-root fact-checking process in BIRDWATCH, including which content is selected to be fact-checked, which sources of evidence are used, and the fact-checking outcome. We also look at possible bias in terms of volume and topics as compared to experts. To enable a fair comparison across the three fact-checking approaches (i.e., computational methods, crowd, experts), we collected a dataset of 11.9k tweets with BIRDWATCH checks and identified 2.2k tweets verified both by BIRDWATCH users and expert journalists. This dataset enables us to analyze and contrast the three approaches across the main dimensions in the standard fact-checking pipeline (see Figure 6.1). We focus on the following research questions:

RQ1 How are check-worthy claims selected by BIRDWATCH users? Can the crowd identify check-worthy claims before experts do?

RQ2 What sources of information are used to support a fact-checking decision in BIRDWATCH and how reliable are they? Does the crowd always rely on data previously fact-checked by experts, or can they be considered as "independent fact-checkers"?

RQ3 Are crowd workers able to reliably assess the veracity of a tweet? Is their assessment always considered helpful by others?

Our results reveal insights from real data to answer these questions. As automatic methods are still not competitive for checking the truthfulness of online content, we focus on how the crowd can fact-check claims and the way they do it as compared to experts. The main contribution of this work is an in-depth data-driven study of how crowd-sourced fact-checking can work in practice, as compared to expert fact-checking over a number of dimensions such as topics, sources of evidence, timeliness, and effectiveness.

Claim Selection. The first step in the process is deciding which claims, out of the very many produced on Twitter, should be fact-checked. This is similar to the process of assessing relevance in a search task, as users are looking for a piece of content that is valuable and that satisfies their requirements (e.g., an information need or potential harm caused by the piece of content if misleading). Regarding the selection of the claims to check, we show that the crowd mostly matches the claims selected by expert fact-checkers in terms of topics, and it is not strongly influenced by properties of the social network, such as popularity of tweets. Moreover, we analyze the responsiveness of crowd and experts with respect to fresh tweets and found that in some cases the crowd is orders of magnitude faster in generating a correct fact-checking outcome.

Evidence Retrieval. In terms of sources, BIRDWATCH users and fact-checkers rely on different set of online resources, with only few reference websites in common. For the sources used by the crowd and the experts, we also compare the quality perceived by the BIRDWATCH community against the quality ratings obtained from a professional journalistic tool. The two scoring methods show correlation, but also remarkable bias in source quality assessment by the crowd on some topics related to politics.

Claim Verification. In terms of effectiveness of the claim verification, we show that crowd-sourcing may be an effective checking strategy in most settings, even comparable to the results obtained by human experts, but does not lead to consistent, actionable results for some topics. We also analyze the agreement among BIRDWATCH users and the use of different scoring functions to aggregate their feedback, including the one used in production by Twitter.

Our observations show how crowd-sourcing fact-checking in practice can bring an added value as compared to expert fact-checkers or computational methods used in isolation. Additionally, we release the first dataset of tweets with labels from expert fact-checker, crowd, and computational methods.

In the rest of the chapter, we discuss related work in fact-checking and crowd-sourcing (Section 6.2), introduce the datasets collected and crafted for our study (Section 6.3), and discuss the empirical results for our analysis (Section 6.4). Finally, we discuss the main challenges and opportunities for crowd-sourced fact-checking (Section 6.5) and conclude the chapter with some open research questions (Section 6.6).

6.2 Background and Related Work

Fact-checking requires a chain of steps that starts with identifying check-worthy claims and ends with a label about the veracity of the claim. Labels vary across services but

usually can be divided into four popular categories: true, partially-true, false, or not enough evidence to judge. The top of Figure 6.1 shows a generic high-level fact-checking pipeline (Nakov et al., 2021a). The three considered checking methods are then reported, specifically automatic methods, BIRDWATCH, crowd, and expert fact-checkers. Given an input textual tweet, every method can be used to assess if it is worth checking and eventually verified. For every checking method, we also report the dimensions that can be used to compare and contrast the alternative methods. We discuss next the main steps in the pipeline, their related work, and their implementation in the different methods.

Claim Selection. For claim selection we can use automatic methods, the crowd, or experts. Given a sentence, an automatic method scores if it contains check-worthy factual claims (Hassan et al., 2017a; Atanasova et al., 2019; Hansen et al., 2019). A model trained on annotated sentences gives low scores to non-factual and subjective sentences. Deciding whether a claim is worth checking is roughly similar to the task of judging the relevance of a document w.r.t. a search query. In Information Retrieval evaluation, well-trained experts (e.g., NIST assessors) may be used to produce judgements of relevance following guidelines, or be instead substituted by crowd workers who receive simple instructions. Similarly, check-worthiness may be performed by a panel of experts or crowd-sourced, like done in BIRDWATCH. The crowd-sourced annotation of textual content on social networks is a widely supported activity across all platforms. Users label content that violates the guidelines of the site, such as hate speech and misinformation. This process triggers the human verification with moderators hired by the platform (Meta, 2021; Dori-Hacohen et al., 2021). For expert, human fact-checkers the selection of the claims to verify is driven by journalistic principles, e.g., claims should contain *verifiable* facts (Holan, 2018). Experts also assess if a claim is *important*, with a definition that changes according to the public and the mission of the organization, e.g., voters and elections (Kessler, 2017). The crowd may have different criteria and priorities in deciding which claims to fact-check and a definition of check-worthiness that takes into account the topic, the timeliness, and their own personal points of view. Previous research in crowd-sourced fact-checking (e.g., (Pinto et al., 2019)) has not looked in detail at how the crowd may perform this step of the pipeline, and it is something that instead we do in this work.

Evidence Retrieval. For computational methods, we distinguish the task of detecting previously fact-checked claims and the task of gathering evidence to support the verification step. As false claims are often repeated across platforms and over time, independently of available fact-checks, claim matching aims at automatically identifying an existing debunking article for the claim at hand (Adler and Boscaini-Gilroy, 2019; Shaar et al., 2020; Botnevik et al., 2020). Claim matching is feasible at scale because websites use the schema.org standard CLAIMREVIEW metadata to share their checks (CRP). For fresh claims, which have not been debunked yet, several methods aim at finding external evidence to help fact-checkers and computational methods deciding on the veracity of a claim (Thorne et al., 2018b). The output is usually a ranking of retrieved documents or specific passages (Alshomary et al., 2020). The crowd makes use of expert fact-checking outcomes when available. Indeed, Roitero et al. (2020a) removed expert outcomes from

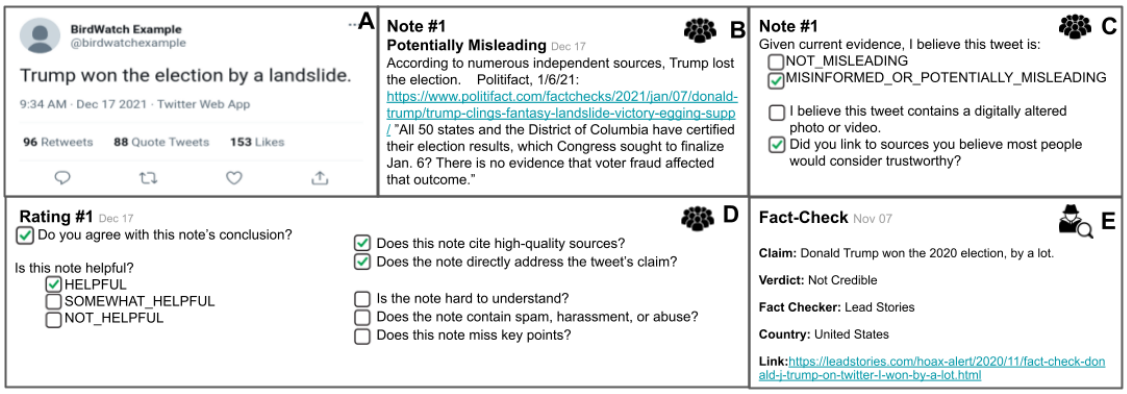

Figure 6.2: BIRDWATCH note and CLAIMREVIEW fact-check Example. (A) shows a tweet. (B) is the note with the assigned label to such tweet. (C) is a sample of questions when submitting a note. (D) is a sample of questions when submitting a rating. (E) shows a fact-check delivered by an expert.

the search results used by the crowd in their fact-checking task to avoid influencing crowd worker judgments. Expert fact-checkers instead rely on their training to identify proven, verified, transparent, and accountable evidence (Rautner, 2020), sometimes involving third-party domain experts (FullFactFAQ, 2022).

Claim verification. A large body of research focus on developing and evaluating automatic solutions for fact-checking (Nakov et al., 2021a; Vo and Lee, 2018; Brand et al., 2021; Botnevik et al., 2020; Pradeep et al., 2021; Su et al., 2019; Karagiannis et al., 2020). However, there are coverage and quality issues with automated systems (Arnold, 2020), and thus a pragmatic approach is to build tools to facilitate human fact-checkers (Vo and Lee, 2018). At the same time, effort in artificially creating rumors and misinformation has been shown to be effective (Huynh et al., 2021). The crowd makes use of evidence from the Web and is influenced by their own personal belief and context (Roitero et al., 2020b; Barbera et al., 2020). Interestingly, when misinformation is identified on social media, users tend to counter it by providing evidence of it being misleading (Micallef et al., 2020). This shows an intrinsic motivation that certain members of the crowd have to contribute to the checking process. An approach for crowd-sourced fact-checking is using tools that surface relevant evidence for their judgement (Fan et al., 2020b). This however comes with the risk of over relying on such tools to make judgements (Nguyen et al., 2018).

Finally, there has been some early analysis of the BIRDWATCH data (Allen et al., 2021b; Pröllochs, 2021), but they focus only on the tweets and notes content, while we rely on the manually aligned expert claim reviews to compare BIRDWATCH results against the best solution in this space. A related study has looked at a Reddit community involved in the fact-checking process using a crowd-sourced approach (Hassan et al., 2019).

6.3 Data

Community-driven fact-checking on Twitter is governed by the Birdwatch initiative (Coleman, 2021b), while fact-checks written by journalists and expert fact-checkers are curated using the ClaimReview schema (CRP). In this section, we describe both datasets and how to match similar claims identified by both parties. Approval from authors' institution research ethics committee to perform this study has been obtained prior to commencing.

6.3.1 Birdwatch

Misinformation on Twitter can be mitigated through the Birdwatch program, where participants can identify misleading tweets and provide more context (Coleman, 2021b). Currently, Birdwatch is only available to participants in the US (thus our analysis has been limited to the US), where users can identify misleading information using two core elements: *Notes* and *Ratings*.

Notes. Participants in the Birdwatch program can add notes to any tweet. Their notes are formed from: (i) a classification label indicating whether the tweet is misinformed/misleading (**MM**) or not misleading (**NM**) according to their judgement, (ii) answers to several multiple-choice questions about their decision (Birdwatch, 2021), and (iii) an open text field where participants can justify their choice of the label and possibly include links to sources that prove their point. An example of a note is shown in Figure 6.2 (B,C). The key data we use from the notes are the following:

- *Classification Label*: Whether the tweet is misinformed (MM) or not (NM) according to the Birdwatch user (Section 6.4.3).
- *Note Text*: the text given by the user with the justification for the label (Sections 6.3.4 and 6.4.1).
- *Timestamps*: time at which the note was written (Section 6.4.1).

Ratings. Participants rate the notes of other participants. Ratings help identify which notes are most helpful. A user rates a note by providing answers to a list of questions (Birdwatch, 2021). An example of a rating is shown in Figure 6.2 (D). Out of these questions, we focus on the following:

- *High-quality Sources:* The user answers the yes/no question 'Is this note helpful because it cites high-quality sources?'. We use this information to assess whether Birdwatch users distinguish credible sources (Section 6.4.2.2).
- *Helpfulness Label:* The user answers the question 'Is this note helpful?'. The possible answers are (i) not helpful, (ii) somewhat helpful, and (iii) helpful. We use this information to compute a helpfulness score for notes (Section 6.4.3).

All Birdwatch notes start with a 'Needs More Rating' status until enough ratings are achieved according to a platform-defined threshold (currently set to 5). Once achieved, these ratings are aggregated and weighted by a 'Rater Score' to compute the 'Note Helpfulness Score'. A higher rater score gives more weight to participants (i) whose notes

ID	Tweet	BW Note		CR Fact		Comment
		Text	Label	Text	Label	
#1	Pregnant women, please don't take this vaccine. https://t.co/4KKlnMIbl7	Updated CDC guidance, and newly accepted and reviewed medical research, has stated there are no safety concerns for pregnant women to be vaccinated against COVID-19. *(links omitted for brevity)*	MM	The vaccine is not safe for pregnant women or women planning on becoming pregnant within a few months of taking the vaccine... We are the lab rats.	NC	The BIRDWATCH user provides proof of why the claim is False. The fetched fact-check has a label of being not credible.
#2	The mass shooting at Marjory Stoneman Douglas High School in Parkland, Florida was real and not staged.	That this continues to be debated is astounding. Yes, this really happened. Here is a link: https://en.m.wikipedia.org/wiki/Stoneman_Douglas_High_School_shooting	NM	Say David Hogg is a crisis actor.	NC	BW note confirms the tweet, thus the label was not misleading. The CR check states a claim opposite to the tweet and its label is not credible.
#3	Chicago PD Says Enhanced Vid Shows Gun in 13-Year-Old Adam Toledo's Hand https://t.co/BOTwu733RL	Chicago Mayor Lori Lightfoot said Adam Toledo had a gun in his hand when he was fatally shot by a police officer, or words to that effect.	MM	Adam Toledo did not have the gun in his hand when he was approached by the police who shot him. He has his arms up and complied. The gun was on the floor, not his hands.	CR	Difference in granularity of the claim. For CR check, the claim is whether Toledo was holding a gun; while for the BW note, the claim was whether Chicago's mayor *said* that Toledo hold a gun.
#4	New poll indicates Biden approval at 11%. The LOWEST approval rating of ANY president in American history. Gallup via Daily Caller	https://fivethirtyeight.com/features/how-were-tracking-joe-bidens-approval-rating/	MM	Biden approval at 11%	NC	The BIRDWATCH participant provided a reliable source (score of 92.5) 7 days before a fact-check by an expert was available.
#5	Biden thinks he came to the US Senate 120 years ago?!?	US President Joe Biden made a clear joke at his first press briefing since his inauguration, in which he said he went to the Senate 120 years ago. This is a self-deprecating joke and shouldn't be taken seriously.	MM	Joe Biden said, 'With regard to the filibuster, I believe we should go back to the position of the filibuster that existed just when I came to the United States Senate 120 years ago.'	CR	The tweeter took a joke seriously, which was interpreted as misleading by the BIRDWATCH participant.

Table 6.1: Examples of tweets, BIRDWATCH Notes, and matched CLAIMREVIEW fact-checks. BIRDWATCH uses labels misleading/potentially misinformed (MM) or not misleading (NM), while CLAIMREVIEW uses credible (CR) or not credible (NC).

are found helpful by other participants, and (ii) whose ratings align with the final rating outcome. A higher note helpfulness score means that many participants found a note adequate, and it would likely hold a valid classification label.
A complete list of questions asked for notes and rating (during the time of writing) are

shown in Appendix D.1.

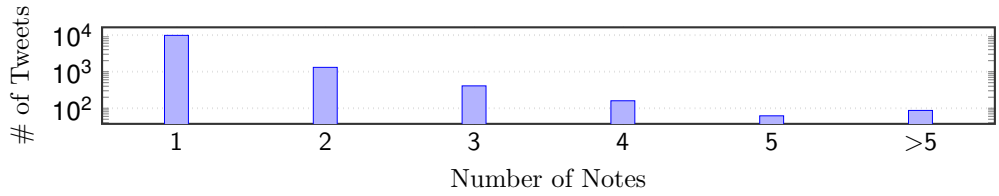

Figure 6.3: Bar plot of the number of notes per tweet.

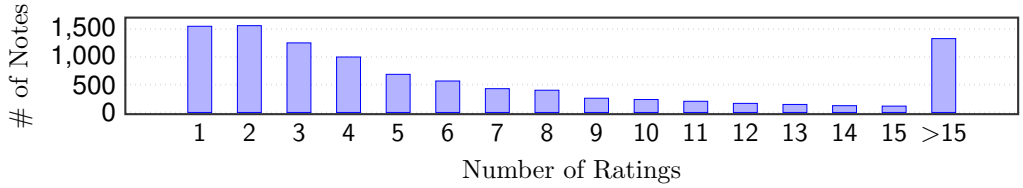

Figure 6.4: Bar plot of the number of ratings per note.

Descriptive Statistics. We use the BIRDWATCH data up to September 18^{th} 2021. The dataset contains 86,924 ratings for 15,445 notes on 11,871 tweets from 5124 unique BIRDWATCH participants. Bar plots of the number of notes and ratings are shown in Figure 6.3 and Figure 6.4, respectively. Most tweets have only one or two notes, while the tweet with the most notes has 61. The majority of notes have less than five ratings, and the most rated note has 184 ratings. The user with most notes checked 656 tweets, with around 71% related to US Politics. Among these 656 tweets, 643 do not have any other note. The user with most notes in common with other users shares 85 notes (on 85 tweets) with 217 other users.

6.3.2 CLAIMREVIEW

The CLAIMREVIEW project (CRP) is a schema used to publish fact-checking articles by organizations and journalists. The schema defines mark-up tags that are used in web pages so that search engines identify the information in a debunking article, such as text claim, claim label, and author (CRS). Our dataset is a collection of items following the CLAIMREVIEW schema, collected from various sources (Mensio and Alani, 2019). Each item, or *fact-check*, is a (claim, label) pair produced by a professional journalist or fact-checking agency. We assume that professional fact-checkers do not overlap with BIRDWATCH participants, as the former have no interest in doing their work without retribution. Since different fact-checkers use different labels, the data is normalized into a smaller subset of labels (credible, mostly credible, uncertain, unverifiable, not credible). In addition to the claim and the label, the checks also contain a link to the fact-checking article. Note that checked claims in this dataset could occur anywhere on the web and

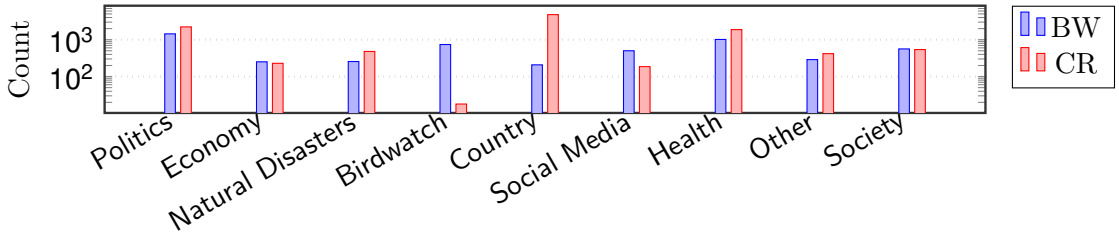

Figure 6.5: Bar plot of tweets checked by BIRDWATCH (BW) and CLAIMREVIEW (CR) fact-checks for 2021 divided by topic.

need not be only on Twitter. We use a dataset containing 76,769 fact-checks. Examples of the data are shown in Figure 6.2 (E) and Table 6.1.

6.3.3 Matched Data

To study how the judgements of the crowd compare to those of expert fact-checkers, we match claims from both datasets. As the automatic matching is imperfect, we used the MTurk crowd-sourcing platform for matching the text in the tweets checked by the BIRDWATCH crowd with the claim text in the CLAIMREVIEW fact-checks. When workers accepted a Human Intelligence Task (HIT), they were shown (i) the tweet that is to be matched and (ii) the top-10 similar CLAIMREVIEW checks provided by SentenceBERT using a bi-encoder with cosine similarity between the text of the tweet and that of the claim in CLAIMREVIEW fact-checks (Reimers and Gurevych, 2019). We also add a 'None of the Above' option for cases where the worker could not find a match. A manual inspection of the matches showed that the vast majority of tweets with a score below 0.6 do not have matching CLAIMREVIEW checks. We therefore run the annotation for tweets with at least 0.6 as top-1 similarity score. The workers were required to have at least 500 approved HITs to access our task, which comprised of 5322 tweets to be matched. Each tweet was shown to 3 workers, similar to previous work (Kazai, 2011; Liu et al., 2010). The hourly rate based on median completion time was 12.41$.

To measure the quality of the worker annotations, we manually annotated the top-500 tweets in terms of matching score. Among these 500 tweets, we manually identified 75 with a matching CLAIMREVIEW check. Workers correctly matched 63/75 tweets according to our ground truth, while the baseline method choosing the highest SentenceBERT score correctly matched 59/75 tweets. After running the study over the 5322 tweets, we obtain 2208 tweets (3043 notes) matching with CLAIMREVIEW checks. An example of a tweet matching a CLAIMREVIEW check is shown in Figure 6.2 (A,E). More examples of matched tweets, BIRDWATCH notes, and CLAIMREVIEW checks are in Table 6.1. Our dataset containing matched tweets to CLAIMREVIEW checks alongside labels from BIRDWATCH and CLAIMREVIEW and relevant code are available at https://github.com/MhmdSaiid/BirdWatch.

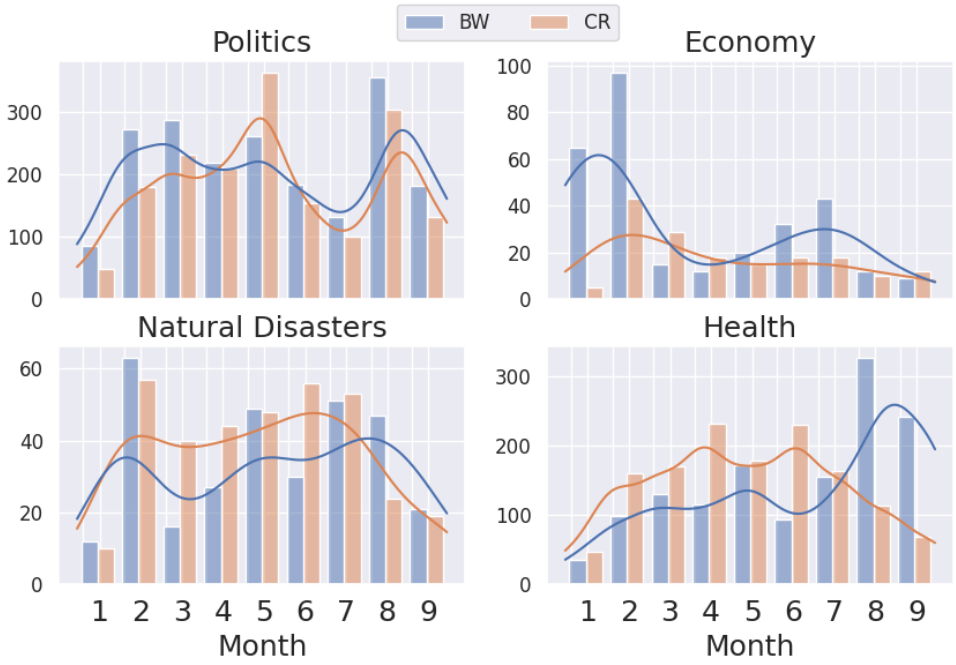

Figure 6.6: Per-topic frequency histograms and KDE Plots for BIRDWATCH (BW) notes and CLAIMREVIEW (CR) fact-checks (month granularity).

6.3.4 Topics

We analyze how BIRDWATCH notes and CLAIMREVIEW checks compare in terms of covered topics. We use *BertTopic*, a topic-modeling technique that utilizes transformers and TF-IDF for clustering (Grootendorst, 2020), to predict the topic of every BIRDWATCH tweet and CLAIMREVIEW claim for the year 2021 and report their frequency distributions in Figure 6.5. *Politics* and *Health* have high counts in both. Topic *Country*, which includes news about countries all over the world, has higher counts for CLAIMREVIEW data since BIRDWATCH is deployed in the US only. BIRDWATCH notes cover mostly tweets in English and is biased towards US related news, whereas the CLAIMREVIEW data contains fact-checks in different languages and from local fact-checking agency, thus explaining the high number of country-related tweets.

6.4 Results

We report results in addressing our three research questions next.

6.4.1 RQ1: Claim Selection

We analyze how BIRDWATCH participants effectively identify check-worthy claims in a comparison with fact-checking experts. We also compare BIRDWATCH users, who do not

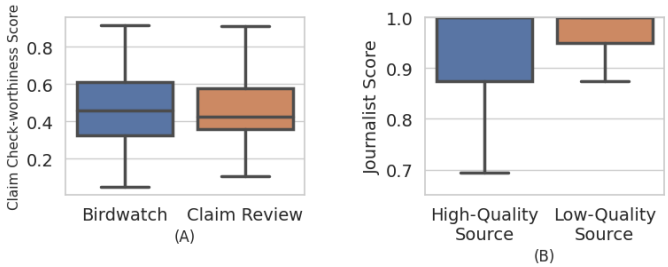

Figure 6.7: (A) shows a box plot of claim check-worthiness scores of BIRDWATCH tweets and the claims in the CLAIMREVIEW fact-checks. (B) shows a box plot of journalist scores compared to the final verdict of BIRDWATCH users (x-axis).

necessarily have journalistic training, against computational methods for this task.

6.4.1.1 Topic Analysis

After predicting the topic of every BIRDWATCH tweet and CLAIMREVIEW fact-check, we plot the frequency distribution of four topics showing interesting trends, on a monthly basis, in Figure 6.6. The high count of BIRDWATCH tweets and CLAIMREVIEW fact-checks covering political tweets show that they both consider the *Politics* topic important. The similar trends for this topic suggest that both methods react similarly to news and major events in terms of claim selection. For example, the peak in *Politics* for both methods in August is related to the Taliban take-over of Afghanistan. We observe the same trend for the topics *Economy* and *Natural Disasters*.

However, for the *Health* related tweets, we observe an abrupt change in the trends from July 2021. This is due to the emergence of the COVID-19 Delta variant in US, which triggered more tweets about the topic, mainly discussing masks/vaccines issues, and more BIRDWATCH notes on this topic. This is accompanied by a decrease in the number of health-related fact-checks, which can be explained by multiple reasons. One explanation is that the most important issues about masks and vaccines had already been debunked before the Delta variant. This shows that despite fact-checks being available online, numerous social network users keep spreading false claims that have previously been debunked (see also Section 6.4.3).

Topic selection also reflects the different geographic focus of the two methods. For example, the BIRDWATCH peak in February in topic *Economy* is due to the Texas power crisis, a US-specific event. Despite the differences, our results show that both BIRDWATCH participants and CLAIMREVIEW experts pick the content to verify in response to the events happening in reality, independently of the specific topic.

6.4.1.2 Computational Methods

We report on the ClaimBuster API for claim check-worthiness (Hassan et al., 2017a). Given a sentence, the API provides a score between 0.0 and 1.0, where a higher score

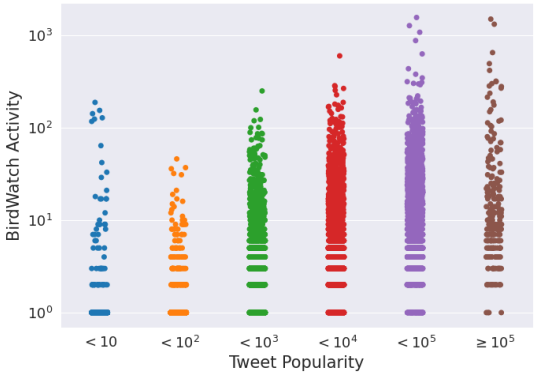

Figure 6.8: Tweet popularity and BIRDWATCH activity.

indicates that the sentence contains check-worthy claims. We run the API on BIRDWATCH tweets and the claims in the CLAIMREVIEW fact-checks, with the associated box plots for the scores in Figure 6.7 (A). The results show a check-worthiness median score at around 0.4, for both sets of claims, while in ClaimBuster the suggested threshold for check-worthiness is 0.5 (Hassan et al., 2017a). One explanation of the difficulties of computational methods for claim selection is the bias in the training data used to build them. Indeed, most available datasets for this task are of high-quality text, coming from articles or political speeches, while the text used on Twitter is usually much noisier, e.g., due to the use of slang.

6.4.1.3 Tweet Popularity

We check whether the claim selection process of BIRDWATCH users is affected by the popularity of a tweet. For every tweet, we retrieve the number of retweets and favorites and sum them to obtain a quantifiable popularity score. As expected, Figure 6.8 shows that popular tweets receive more activity than others from the BIRDWATCH community, i.e., have more notes and ratings. However, there are popular tweets with low BIRDWATCH activity and unpopular tweets with high number of notes and ratings.

6.4.1.4 Temporal Analysis

We analyze tweets (T), BIRDWATCH notes (B), and CLAIMREVIEW fact-checks (C) time-wise. As a note can only occur after a tweet, we have three different configurations: (i) Tweet occurs first, then BIRDWATCH note, then CLAIMREVIEW fact-check (TBC), (ii) Tweet then CLAIMREVIEW fact-check then BIRDWATCH note (TCB), and (iii) CLAIMREVIEW fact then Tweet then BIRDWATCH note (CTB).

TBC: There are 129/2208 tweets in our matched data for this case. In all tweets, BIRDWATCH users provide a response much faster than experts. On average, a BIRDWATCH provides a response 10× faster than an expert. These examples show how BIRDWATCH

participants can fact-check claims with reliable sources without the need of ClaimReview fact-checks such as ID #4 in Table 6.1.

TCB: In our dataset, a ClaimReview rarely occurs after a tweet and before a Birdwatch. We observe faster responses from ClaimReview than Birdwatch users for 26/2208 tweets. Since the granularity of the ClaimReview is days while that of Birdwatch is seconds, there are also 17/2208 tweets that occur on the same day, and we cannot state which of the two was actually faster.

CTB: The majority of the matched tweets follow this pattern, with most of them related to US politics and COVID-19. As Twitter is an open space, several users tend to spread false news even after they have been fact-checked, specifically those related to Trump winning the elections. We discuss more this issue in Section 6.5.

Claim Selection Take-away Message: Birdwatch users and ClaimReview experts show correlation in claim selection decisions w.r.t. major news and events, but with important differences due to the circulation of claims that have been already debunked by experts. The crowd seems to be effective also in identifying tweets with misleading claims even before they get fact-checked by an expert. Also, both popular and non-popular tweets get verified by Birdwatch users. Computing the check-worthiness of a tweet does not lead to effective results using current off the shelf APIs.

6.4.2 RQ2: Evidence Retrieval

Both crowd checkers and experts report the sources used in their verification process. We analyze such sources and then contrast their quality according to an external journalistic tool[1].

6.4.2.1 Descriptive Statistics

We extract all links from Birdwatch notes. We find a total of 12,909 links covering 2,014 domains. Unsurprisingly, the top cited links are those coming from journalistic and fact-checking sites (PolitiFact, Reuters, NYtimes) and governmental websites, such as USGS and CDC. The distribution of the links is right-skewed, where half of the links are from only 29 domains. ClaimReview checks contain 76,769 links covering only 73 domains of fact-checking groups and journalists. The distribution of links shows less skewness than that of Birdwatch. Birdwatch participants use only 17 domains in common to those of the ClaimReview experts. The other 56 ClaimReview domains, which are not in the overlap, include 53 local resources, such as news outlets, for non US countries as fact-checking organizations work at a global scale and Birdwatch focuses on US. Birdwatch sources are a larger number as they range from Wikipedia and YouTube videos to medical websites and research papers.

[1] https://www.newsguardtech.com/

6.4.2.2 Expert Judgement of Source Quality

We compare ratings of source quality of BIRDWATCH users to those of expert fact-checkers. To assess the quality of web sources, we rely on an external tool that provides a score (between 0 and 100) where the higher the score, the higher the quality of the source[2]. The score is obtained by journalists manually reviewing every website, and we refer to it as the *journalist score*.

For every note in our matched dataset, we first compute a BIRDWATCH score indicating whether the links are high-quality sources or not by performing a majority voting on the ratings of the note, and we then compute the average journalist score of every link in the note. Out of 3043 notes, 2231 contained links. We obtained results for 656, while the others either had (i) no ratings (363/2231), (ii) no journalist scores (698/2231), (iii) nor both (309/2231), or (iv) there was no majority in the ranking votes (205/2231).

A box plot of journalist scores and BIRDWATCH labels is shown in Figure 6.7 (B). For note links rated as high-quality by BIRDWATCH users (with majority voting), we observe high journalist scores. The majority of tweets of the notes are related to US elections and COVID-19, with BIRDWATCH users citing sources such as PolitiFact and CDC. Some sources in the notes have been classified as being high-quality by BIRDWATCH users, but low-quality w.r.t. the journalist scores. Those notes share mainly COVID-19 studies such as Mayoclinic.org, a nonprofit American medical center, and fda.gov, the US food and drug administration, that are regarded as reliable sources in the US but do not meet all the requirements for high journalist score.

238/656 notes contain sources that are rated as low-quality, but have a high journalist score. These notes are debunking news about US politics, specifically about Trump winning the 2020 elections, and misinformed COVID-19 content. These tweets include links to reliable sources, but a significant fraction of BIRDWATCH users labeled such links as low-quality. This shows how some BIRDWATCH users convey partisanship, forming a group of people trying to deceive the BIRDWATCH program to serve their common interest, such as supporting a political party in social media.

Such groups can be effective in "gaming" the algorithm (Epstein et al., 2020), ultimately having a profound effect on BIRDWATCH since (i) the biased BIRDWATCH participants can steer the ultimate label of a note to their favor, thus spreading misinformation, and (ii) by increasing their weight in the BIRDWATCH platform since if one's ratings match those of the ultimate rating, they will get a higher weight. As an example, for the tweet `'Joe Biden is President In Name Only. #PINO'`, a certain note replied that Biden is indeed the president with links from *PolitiFact* and *APNews*, both having journalistic scores of 100/100 and 95/100 respectively while 12/14 of the raters identified such sources as unreliable.

We also compute journalist scores for links in BIRDWATCH and CLAIMREVIEW data. As BIRDWATCH users use many links, we only computed scores for the top-100 occurring links that form 68.6% of the data. While both distributions of BIRDWATCH and CLAIMREVIEW link scores attain a median of 1.0, links by CLAIMREVIEW fact-checks

[2] https://www.newsguardtech.com

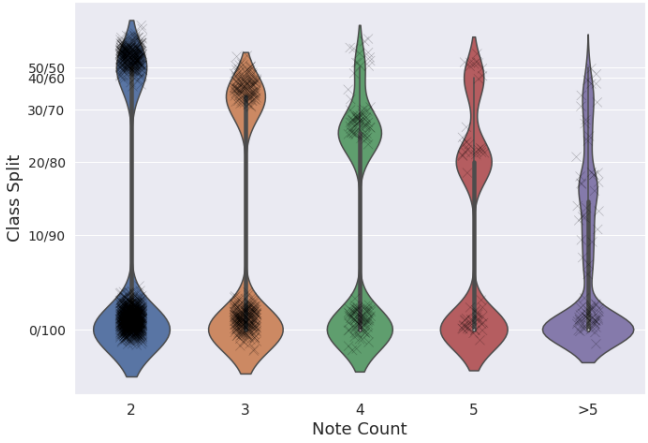

Figure 6.9: Violin Plot of note counts and class splits of the classification labels of notes. The figure shows the Kernel Density Estimation (KDE) plot of every note count, with their respective data points. Tweets with two notes are dominant, with most notes agreeing on the label. Other note counts do show full agreement on the label (0/100), with more cases of disagreement as the number of notes increase.

have lower variance with a minimum of 0.875, while that of BIRDWATCH notes is 0.495.

Evidence Retrieval Take-away Message: Expert fact-checkers rely on a relatively small set of high-quality sources to verify claims, while BIRDWATCH participants provide a variety of sources that seem to be neglected by fact-checkers. While most of these sources are evaluated as credible (by journalists) and useful (by the BIRDWATCH crowd), malicious users might game the algorithm and effectively label notes as unhelpful according to their ideology.

6.4.3 RQ3: Claim Verification

We ponder whether BIRDWATCH participants provide accurate judgements. We first compare agreement (i) among themselves and then (ii) with CLAIMREVIEW expert fact-checkers. We then analyze different scoring functions for note aggregation, and finally report results for computational methods.

6.4.3.1 Internal Agreement

We use the participants' classification labels to see whether the tweet is classified as misinformed or not. To compute agreement, we use the standard metrics Krippendorff's alpha (Krippendorff, 2011) and Fleiss's kappa (Fleiss, 1971). However, due to the large sparsity in the data and the huge number of missing values, both metrics fail to provide meaningful results (Checco et al., 2017). We then compute the variance as a metric for agreement. Lower variance means that all BIRDWATCH participants agree on the

		Birdwatch				
		Notes		Tweets		
		MM	NM	MM	NM	Tie
Claim Review	credible	209	**25**	126	**9**	9
	mostly_credible	56	**14**	44	**7**	5
	not_credible	**1983**	184	**1476**	62	55
	not_verifiable	300	25	225	8	9
	uncertain	225	22	156	8	9

Table 6.2: Matching the classification labels across Birdwatch and ClaimReview on the note level and the tweet level (obtained through majority voting). Agreement in bold.

classification label. A violin plot is shown in Figure 6.9 for tweets with various note counts. On the y-axis, we report the density of the class splits, where a class 0/100 indicates full agreement across the users and 50/50 indicates full disagreement. We see that most tweets have two notes and the majority of users perfectly agree on the final classification label. The same applies to tweets with more note counts, where most of the notes agree on the final label, with conflicts happening on some tweets but with a small subset with full disagreement. A topic analysis of tweets shows that 48.3% of tweets with full disagreement are related to either politics or COVID-19.

6.4.3.2 External Agreement

After matching Birdwatch data with ClaimReview fact-checks, we compare their labels. Table 6.2 shows that the majority of ClaimReview labels match the Birdwatch ones. Specifically, in terms of notes, there are 2022 cases (25+14+1983) where they agree and 449 cases of disagreement. In terms of tweets, there are 1492 (9+7+1476) decisions with the same classification label and 232 (126+44+62) with different labels. For 69 (9+5+55) tweets, there is a tie in the voting across Birdwatch users. For completeness, we report also the numbers for other ClaimReview labels ('not_verifiable' and 'uncertain'), even if they have no mapping to Birdwatch classification labels. We did some analysis to understand the cases where the labels are not aligned. Some examples are reported in Table 6.1. Among the 209 notes that are labeled as credible by the ClaimReview fact-checks and misinformed by the Birdwatch participants, the most common cause are texts with multiple claims, i.e., multiple facts are reported in a tweet and the fact-checked claims differ (ID #3). In other cases, tweets are mistakenly labeled as misinformed, e.g., because a joke is taken seriously by a Twitter user (ID #5). Finally, assuming correct ClaimReview labels, we believe in some cases the mismatch is due to biased Birdwatch users. For the tweets labeled as not credible by ClaimReview fact-checks and not misleading by Birdwatch notes, we observe cases where a Birdwatch note is the negated version of the ClaimReview fact-check (ID #2), thus producing opposite labels. There are also mismatch of labels, even though the Birdwatch user provides evidence from a link that has a high journalistic score (0.875).

6.4.3.3 Note Helpfulness Score

In the real-world production setting, not all BIRDWATCH notes are used for finding the ultimate label that gets exposed on the platform. In fact, a *note helpfulness score* is computed by the platform for each note, and those having a high enough score are used for computing the ultimate label. BIRDWATCH exposes the code for computing such score, however, the public code does not include raters' scores into consideration. We use the available code and filter out notes that are not helpful for the final label, using a Twitter-defined threshold for the note helpfulness score (0.84). We are left with 533 tweets (over 2208) that pass the threshold, with 333 notes labeling the tweets according to CLAIMREVIEW checks. About 95% of notes label the tweets as misleading, thus indicating that BIRDWATCH users tend to rate misleading tweets more than non-misleading ones, in agreement with previous work (Pröllochs, 2021). Of course, malicious ratings of the classification labels can steer the note helpfulness score in misleading directions, similarly to the judgement for source quality as discussed in Section 6.4.2.2.

6.4.3.4 Computational Methods.

We compare our matched data with labels coming from computational fact-checking systems. The systems discussed in Chapters 3 and 4 are not directly usable as they require specific labeled data, which is not available from the notes. Moreover, such systems handle subclasses of claim types that do not cover the type of claims found in the majority of the tweets. Most claims found in tweets require finding evidence related to very recent events, which is not available in databases, knowledge graphs, or PLMs, and also require human-reasoning capabilities. Nevertheless, we use other systems such as ClaimBuster, as it can also verify claims (Hassan et al., 2017b), and E-BART (Brand et al., 2021). ClaimBuster provides correct results for 118 out of 2208 tweets, where 2090 tweets have no output from the model with an F1-score of 0.042. E-BART correctly labels 369 (over 2208) and does not produce a decision for 59 tweets with an F1-score of 0.17. A random classifier produces an average F1-score of 0.333 with 0.008 standard deviation. As for claim selection, tweets are harder to handle for computational methods than news articles and quotes from politicians, which are the bulk of content in training corpora.

Claim Verification Take-away Message: BIRDWATCH users show high enough levels of agreement to reach decisions in the vast majority of cases. The BIRDWATCH crowd focuses mostly on misleading tweets and shows high agreement with expert fact-checkers in terms of classification label. Computational methods have room for improvement in automatically verifying tweets.

6.5 Discussion

6.5.1 Collaborative solutions

The analysis of the quality of the BIRDWATCH users shows that crowd-sourced fact-checking is a promising and complementary solution, with results that correlate with those of professional fact-checkers. However, we argue that a crowd-based solution should not be considered to replace experts, but rather as a tool in collaborative effort where, for example, the crowd helps flagging content and creating links to more sources of trustable evidence. Indeed, our results show that the crowd can be even more reactive than experts to a new false claim and is able to identify a large array of high quality sources of evidence. This is especially important, as there is evidence that fact-checking interventions are significantly more effective in novel news situations (Nevo and Horne, 2022).

Looking forward, and assuming we can characterize the trust and the cost level for all involved actors (i.e., crowd, experts, and computational methods), there is an opportunity to design novel hybrid human-machine solutions that coordinate this joint effort in order to combine the benefits of the different approaches. The role of automatic tools can be that of providing real-time and scalable fact-checking for all posted content. Platform users can then intervene quickly in a more focused manner to provide a first line of defense on potentially harmful content. This can then be followed by quality in every step of the fact-checking pipeline, with humans collectively processing evidence for the final labeling.

6.5.2 Hard-to-verify claims

The matching process of tweets and claim-review checks led us to recognize the difficulty of this task. The first challenge is the semantic match in terms of content, but in many cases where the match is clear, the problem is hard even for humans. Several problems, such as sarcasm and vagueness, are known in general for the detection of worth-checking claims (Atanasova et al., 2019). One possible solution might the incorporation of a crowd-sourced "tipline" through which users can flag the content to be fact-checked (Kazemi et al., 2021). However, another problem is the granularity of the tweet. Even a very short tweet may contain two interleaved claims, such as "Mike said: the earth is flat" (see also, e.g., tweet #3 in Table 6.1). Assume there are two claim reviews, one checks that Mike made a claim about the earth (labeling the matching tweet as true), and the other checks about the fact that the planet is not flat (labeling the tweet as false). This suggests the challenge of being able to identify the textual claims where both crowd-sourcing and computational fact-checking methods are most likely to fail short. This can be modeled as a new supervised classification task aiming at predicting when a claim cannot be verified effectively without experts. As an orthogonal approach, this is also an opportunity for automatic controversy detection methods (e.g., (Dori-Hacohen et al., 2016)) to play a complementary role in supporting the crowd, making them aware of potential controversies during their verification tasks.

6.5.3 Stale claims

We found clear evidence that claims that have been already verified by expert checkers keep circulating and spreading on Twitter, even months after the publication of their debunking (Shin et al., 2017). Unfortunately, we have also observed that automatic methods still fail short in matching with high accuracy tweets that contain such "stale" claims, likely because of the peculiar language used in tweets. In such a setting, Birdwatch users can play an important role in quickly and effectively recognizing these cases. Indeed, the significant difference in the health-related Birdwatch notes and ClaimReview fact-checks is explained by the increase of tweets spreading already fact-checked claims. Stale claims are a good fit for the role of Birdwatch participants, especially when automatic matching methods (Shaar et al., 2020; Ahmadi et al., 2022b) fail, while fresh claims might require proper forensic processes and need the expertise of journalists.

6.6 Conclusion

As computational fact-checking systems cover only a subset of claims for verification and require costly labeled data, crowd-sourcing approaches offer more coverage of claims. Also, such crowd-based approaches do not require training data, making the verification procedure of a claim, especially a critical one, much faster rather than systems that have to collect data and re-train. In this chapter, we presented a data-driven analysis of the Birdwatch program through the lens of the three main components of a fact-checking pipeline: claim detection based on check-worthiness, evidence retrieval, and claim verification. This is also the first study that bridges real data from a large-scale crowd-sourced fact-checking initiative with the debunking articles produced by professional fact-checkers.

However, such crowd-sourcing approaches have their downsides. A prominent one from our study is that some malicious sharing a common goal could potentially steer the system to their favor, such as in the case of source credibility for Birdwatch. This suggests that more attention is needed in identifying harmful groups by profiling their activity and by incorporating their biases in the note ranking system.

7 Conclusion & Future Directions

In this thesis, we have seen how to extend PLMs to render them more functional for consumers of fact-checking systems. We propose extensions that (i) help the model deal with tables with optimally integrated domain expert feedback for claim verification, (ii) emulate soft reasoning over language, and (iii) query with a specified output type. In addition, we also analyze a large-scale crowd-sourcing program at Twitter. Nevertheless, a lot of obstacles are left and need to be tackled.

7.1 Summary of the Contributions and Future Work

This thesis makes the following contributions:

- **Domain-Specific Fact-Checking with Tabular Evidence and Humans-in-the-Loop** [Chapter 3]
  In this chapter, we propose the system Scrutinizer to verify statistical claims in a domain-specific setting. As input claims are expressed in natural English, and the means to validate them are stored in relational tables, we cast the verification procedure as a Text2SQL problem, where specific query properties are predicted using fine-tuned PLMs, and then queries are generated using a slot-filling approach (Wang et al., 2020; Yu et al., 2018) and executed to verify or debunk the claim in question. However, as our approach is data-driven, labeled data is needed. This becomes infeasible, especially in peculiar domains where experts would have to indulge in a costly labeling task. To tackle this, we propose an active learning procedure to identify a batch of claims that are to be labeled by fact-checkers, and then used for training the models. This allows the models to reach acceptable accuracy with a minimal number of examples.

  Future Work: With the incorporation of tables within PLMs, come many unanswered questions. Tables by themselves contain a rich structure with semantically meaningful attributes. One question is how can PLMs correlate between attributes

that, while are semantically identical, are represented differently. For example, given an attribute about distances in meters, and another about distances in yards, PLMs identifying such attributes as "similar" would require further extensions than standard pre-training in a self-supervised fashion. Such sort of identification is crucial for fact-checking as multiple tables might be needed for verification (Aly et al., 2021; Karagiannis et al., 2020), especially when tables might originate from heterogeneous sources (Dong et al., 2019). This encourages incorporating data curation techniques within deep learning models (Thirumuruganathan et al., 2020; Cappuzzo, 2022), specifically for tables with PLMs, which have not received much attention. Another important hinder, especially for Text2SQL models, is the inability to perform complex queries with aggregations and joins. As most work assumes a single table as input, developing models that handle multiple tables is inevitable for established claim verification.

- **Emulating Soft Reasoning with PLMs** [Chapter 4]
  In this chapter, we propose RuleBERT as a methodology to emulate reasoning over language with PLMs. Specifically, our proposed data-generation algorithm allows handling rules with binary predicates rather than unary ones. We also propose to incorporate the weight of the rule in the objective function as a means to handle the uncertainty of rules. Such rules that are both soft and containing binary predicates allow for better representation of knowledge in real-world settings (Russell and Norvig, 2010). This is also essential in the claim verification procedure of a fact-checking system, as 'fuzziness' of the input is crucial to encode degrees of truth of claims. This is shown by applying RuleBERT to external datasets, showing improvement in terms of deductive reasoning and logical consistency in terms of negation and symmetry. RuleBERT could, ideally, be used for logical deductive reasoning tasks (Sanyal et al., 2022) and extended for other logic-related tasks (Clark et al., 2019a; Geva et al., 2021a).

  Future Work: While a line of work tries to 'incorporate' logical rules in neural networks (Li and Srikumar, 2019; Hu et al., 2016; Xie et al., 2019; Clark et al., 2020), obtaining systems that can actually reason is far-fetched. In fact, such systems are more situated into the so-called *System 1* category in deep learning, which are systems that are fast, unconscious, and employ no planning or reasoning; as opposed to *System 2* that are slow, logical, and involve reasoning component(s) (Kahneman et al., 2020). Moving from *System 1* to *System 2* would require multiple steps. One of these steps is the ability to express high-level semantic representations and manipulate them. Examples of high-level semantic representations in logical settings would be predicates such as `spouse` and `occupation`. However, it is not clear whether reasoning with such concepts should happen within the neural network itself (Clark et al., 2020; Saeed et al., 2021a), or through a more hybrid approach where the network interacts with a symbolic system such as a reasoner (Manhaeve et al., 2018; Weber et al., 2019). The former would need a certain methodology to produce a symbolic description of the network for explainability purposes. The latter

would need a way to interface between the neural network and the symbolic system. A more thorough discussion can be found in (d'Avila Garcez and Lamb, 2020). Another main drawback concerning RuleBERT and similar models is catastrophic forgetting (Kirkpatrick et al., 2017; Ke et al., 2021b). As we further fine-tune the model on a new task, we would eventually degrade the model performance on other tasks (such as QQP in Table 4.7). This motivates incorporating continual learning approaches (Wu et al., 2022) to alleviate the effects of catastrophic forgetting.

- **Output Type Specification in PLMs** [Chapter 5]
  In this chapter, we enforce the type of the output prediction of a PLM. This is a vital component for fact retrieval, that implicitly exists during retrievals from KGs, but non-existent when querying PLMs. For this, we develop TEs, additional input embeddings that encode the desired output type. We devise a simple method to derive the TE, and then follow with techniques to analyze it. Finally, we run experiments on the LAMA dataset, showing improvements with TEs. We observe that fact retrieval from PLMs can be enhanced by integrating type information. Our experiments show that, with as a few as ten labeled typed-tokens, one can encode the type and inject it in the model without the need of external resources or further training of the model, agreeing with other work showing that latent concepts, and hence types, are encoded by PLMs (Dalvi et al., 2022).

  Future Work: Controlling the output type of PLM is a vital component of triplet-fact retrieval that arises with the transition from structured systems such as KGs to PLMs. Controlling the type can help in many scenarios such as efficient information retrieval for fact-checking (Petroni et al., 2019), or rule mining from PLMs (Cui and Chen, 2021). TE is one approach for doing so, however, it comes with its limitations. As we turn to PLMs for deriving the TE, we are restricted by the tokens in its vocabulary, which limits the number of possible types for TEs. One possible remedy is to integrate modules that adapt a PLM's vocabulary to new unseen tokens (Chen et al., 2022; Hong et al., 2021). In addition, while we use TEs for a factual dataset, the TE encodes only type information and no factual information. While results improve for LAMA with TE, the interaction of type information and factual knowledge of the PLM is not fully understood yet. A series of works have tried to understand how factual information emerge during PLMs (Meng et al., 2022; Geva et al., 2021b), and a recent work shows that types such as `language` and `organization` emerge through PLMs such as BERT (Dalvi et al., 2022); however, the study of entanglement of both factual and type knowledge for fact retrieval is still lacking. A possible route would be to investigate the key-value memories in a PLM, as some values might be able to encode the desired type (Geva et al., 2022). Finally, it would be interesting to understand how TEs behave in other scenarios, such as text generation. A preliminary experiment shows that types can be reflected in the output text without any fine-tuning of the PLM, but a more in-depth study should investigate how TEs behave in a "plug-and-play" controlled text generation scenario (Dathathri et al., 2020; Pascual et al., 2021).

- **Analysis of a Large-scale Crowd-sourcing Initiative** [Chapter 6]
  In this chapter, we lay our attention on the BIRDWATCH initiative, a program launched by Twitter to combat fake news through crowd-sourcing. We perform analysis regarding claim selection, evidence retrieval, and claim verification by processing 11.9k tweets. We compare the results of BIRDWATCH users against themselves and against expert tools and fact-checkers. We also release a dataset of matched tweets and BIRDWATCH notes with expert fact-checks. Our work goes in line with work showing that the wisdom of the crowd can be effective in terms of claim verification (Allen et al., 2021a; Resnick et al., 2021; Bhuiyan et al., 2020), as the crowd can be much faster to verify or debunk a claim than expert fact-checkers. However, malicious groups serving s specific purpose can manipulate the program in their favor. Such coordinated manipulation attempts are currently addressed through certain eligibility criteria that a BIRDWATCH participant should follow.

  Future Work: While the BIRDWATCH initiative takes a vital step towards crowd-sourcing based approached for fact-checking in uncontrolled environments, more efforts should be carried in the behavioral analysis of users *per se*, and how they interact with other groups of users, eventually identifying cliques that follow similar behavior (Chakraborty et al., 2017; Dhawan et al., 2019). Also, there might be an opportunity to extend the program with solicitations to experts that optimize a certain objective. This objective would ideally minimize verification costs by consulting experts in demanding onerous scenarios, such as tweets having high conflicts between the participants themselves, or tweets showing unusual activity, like participants labeling links of fact-checking organization as unreliable. This might help identify the bad players and alleviate their influence. Another non-trivial aspect is the difficulty to capture the intricacies when fact-checking a claim. Determining the veracity of a claim is not as challenging as detecting the level of harmfulness of a tweet, or whether more context is needed for verification. Such 'meta-tasks' of fact-checking cannot be modeled with the same perspective used for the current tasks of fact-checking, and require a broader vision.

7.2 Where Do We Stand from Automated Fact-Checking Systems Today?

As numerous academics, journalists, and organizations have indulged heavily in the development of automated fact-checking (AFC) systems, one cannot but emphasize the vital role technology has offered in helping fact-checkers. However, it is crucial to take a step back and reflect on what technology has not (yet) offered for automated fact-checking:

- **Trustworthiness of (Heterogeneous) Sources.** Current systems usually rely on a single source of information for assessing the validity of a claim. However, "the most dangerous misinformation for each of us comes from the sources we trust". While sources are key in fact-checking, models should be 'aware' that sources could

be mistaken, and should not heavily depend the decision on information from the source. One might consider including multiple sources rather than a single one; however, not all information is equally trustworthy, and sometimes sources contradict each other. Systems should be able to model sourcing being contradicting, or not fully trustworthy. Soft logic approaches (as the one used in Chapter 4) can help on passing this degree of truthfulness of the system towards claim verification. Also, care should be taken when sources are of heterogeneous formats, such as knowledge graphs, textual corpora, and relational databases. Computing the source weights should be agnostic of the type of the source. Possible directions could relate with work that tries to assess user reputation in crowd-sourcing (Jagabathula et al., 2014) or social networks (Wang et al., 2012) or 'truth discovery' methods that resolve conflicts among multi-source noisy data (Li et al., 2016).

- **Effective Judgement.** Much of the scope of claims covered by human fact-checkers requires a kind of judgement and analysis of context that remains far out of reach for fully automated verification. Such reasoning capabilities are beyond the capabilities of current AI models now. Nevertheless, much more effective judgement could be attained by taking into consideration the fact that claims could be unverifiable and/or ambiguous. Unverifiable claims arise when there is no sufficient evidence to verify or refute a claim. Currently, unverifiable claims are usually handled by introducing a NOT ENOUGH INFO (NEI) label (Thorne et al., 2018a). Other approaches include leaving out sentences or tables from gold evidence chains to augment NEI examples (Malon, 2021), and predicting the NEI class if there is no information, based on similarity and voting heuristics, that the extracted evidence supports or contradicts the claim (Temiz et al., 2021). One could borrow from the line of work of SELECTIVE PREDICTION that enables systems to abstain from making predictions when they are likely to be incorrect (Hendrycks and Gimpel, 2017; Varshney et al., 2022b,a). Such an approach could also be used for claims incurring some form of ambiguity, resulting in uncertainty among humans (Pavlick and Kwiatkowski, 2019). Approaches to resolving ambiguity, usually include modeling it in the training dataset (Veltri et al., 2022; Meissner et al., 2021; Chen et al., 2020a).

- **Human and PLM Biases.** Employing humans in AFC systems, especially in crowd-sourced fact-checking applications, includes therefore human-imposed biases. This is well studied in the literature: when humans see a claim that does not align with their political views, they are less likely to provide an objective evaluation (Jakesch et al., 2020). Human bias could also re-surge from crowd-sourced datasets (Eickhoff, 2018; Hube et al., 2019; Liu et al., 2022a). Models trained on such datasets tend to rely on their encoded biases, rather than learning the underlying task (Poliak et al., 2018; McCoy et al., 2019; Zhou et al., 2021). While it is not easy to detect such biases, methods of analyzing activity of users could alleviate such bias. Possible solutions include modeling annotator bias probabilistically for achieving more accurate labeling (Wauthier and Jordan, 2011; Liu et al., 2022a) or a

game-theoretic incentive scheme to counter the effect of worker bias (Faltings et al., 2014). On the other hand, PLMs themselves encode biases (Bender et al., 2021), whether stereotypical or negative sentiments towards specific groups. Such biases should be taken into consideration when developing AFC systems, as they affect the system's final decision (Speer, 2017). TEs (Chapter 5) have shown promising results on text detoxification, and are worth studying for reducing bias. Work on debiasing PLMs is indeed gaining traction (Guo et al., 2022a; Lauscher et al., 2021; Khalifa et al., 2021) and future work needs to prioritize debiasing methods that are training-efficient, i.e., do not require full fine-tuning of the model, and are generalizable over multiple NLP tasks, i.e., not just NLG.

- **Multimodality and Multilinguality.** Even though most systems focus on some form of triple or unstructured text, misinformation and reference data could appear in various formats (e.g., images, videos, sound). While most of the work focuses on textual input, fact-checking of multimodal claims remains underexplored in comparison. Some relatively recent work putting on other modalities (Singhal et al., 2019; Alam et al., 2021), however, several challenges persist. One of such challenges is the methodology to combine the different modalities as ideally, one modality should complement the other rather than fusion-based techniques that do not offer so (Zhou et al., 2020a; Zhang et al., 2019). Also, more traction is needed for developing truly multimodal datasets. Recent work has been studying multimodal transformers with attention-based approaches to combine modalities (Xu et al., 2022). Another aspect worth shedding light on is the fact that claims could appear in multiple languages, while most work is biased towards English. Indeed, recent work addresses this by introducing a multilingual benchmark for fact-checking (Gupta and Srikumar, 2021). Meta's "No Language Left Behind" project aims at creating datasets and models that narrow the gap between low and high-resource languages (NllbTeam et al., 2022).

- **Change in Paradigm.** Nowadays, with the stampede of work in AFC systems, both researchers and practitioners agree that the real promise of AFC technologies lies in tools to assist fact-checkers to identify and investigate claims, and to deliver their conclusions as effectively as possible. Nevertheless, it is still crucial to establish some common grounds between researchers and journalists in the field to alleviate some of the ambiguities and hurdles in the fact-checking procedure in order to achieve the next milestones. Interestingly, it seems that aside from all what technology has to offer, the next steps to automate fact-checking might require looking at the problem from a different lens. (Ammara et al., 2020) propose to analyze misinformation through the lens of *Systems Thinking*, a highly-developed discipline that aims to model systems as wholes and relationships between them, rather than single modules (Stroh, 2015). For example, the authors envision that some notions of Systems Thinking can be used to model the reliability of news sources and improve ethical decision-making around misinformed news. (Vlachos, 2020) models fact-checking as a conversation. While conversations might help in broadening one's

point of view around a subject, they could increase polarization (Bail et al., 2010; Balietti et al., 2010). The authors promote the idea of conversations, as reasoning has been shown to work better in groups (Sperber and Mercier, 2018; Laughlin et al., 2003). This could be accomplished by bots that, unlike chatbots, engage in a conversation with a user, without any task-oriented purpose. Other approaches include an 'infodemiological' approach where users study the spread of news the way an epidemiologist studies the spread of diseases (Daley and Kendall, 1964), and philosophical approaches such as that of Ludwig Wittgenstein that could provide a framework for effective misinformation analysis (Omoregie, 2021).

Finally, all advances in AFC systems are futile as long as the freedom of journalists is jeopardized and controlled by malicious parties[1]. Fact-checking is a key-stone in journalism that needs to be preserved under all circumstances. "You have to start with the truth. The truth is the only way that we can get anywhere. Because any decision-making that is based upon lies or ignorance can't lead to a good conclusion." #FreeAssange

[1] https://assangedefense.org/

Bibliography

Claimreview project.

Claimreview schema.

Serge Abiteboul, Luna Dong, Oren Etzioni, Divesh Srivastava, Gerhard Weikum, Julia Stoyanovich, and Fabian Suchanek. 2015. The elephant in the room: getting value from Big Data. In *Workshop on Web and Databases (WebDB)*, Melbourne, France. ACM Press.

Ben Adler and Giacomo Boscaini-Gilroy. 2019. Real-time Claim Detection from News Articles and Retrieval of Semantically-Similar Factchecks. In *NewsIR'19 Workshop at SIGIR*.

Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. 2002. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16.

N. Ahmadi, H. Sand, and P. Papotti. 2022a. Unsupervised matching of data and text. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1058–1070, Los Alamitos, CA, USA. IEEE Computer Society.

Naser Ahmadi, Joohyung Lee, Paolo Papotti, and Mohammed Saeed. 2019. Explainable fact checking with probabilistic answer set programming. In *Truth and Trust Online, TTO*.

Naser Ahmadi, Hansjörg Sand, and Paolo Papotti. 2022b. Unsupervised matching of data and text. In *Submitted to ArXiV, 16 December 2021, Accepted at IEEE ICDE 2022*.

Naser Ahmadi, Thi-Thuy-Duyen Truong, Le-Hong-Mai Dao, Stefano Ortona, and Paolo Papotti. 2020. RuleHub: A Public Corpus of Rules for Knowledge Graphs. *ACM J. Data Inf. Qual.*, 12(4):21:1–21:22.

Firoj Alam, Stefano Cresci, Tanmoy Chakraborty, Fabrizio Silvestri, Dimiter M. Dimitrov, Giovanni Da San Martino, Shaden Shaar, Hamed Firooz, and Preslav Nakov. 2021. A survey on multimodal disinformation detection. *ArXiv preprint*, abs/2103.12541.

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36.

Hunt Allcott, Matthew Gentzkow, and Chuan Yu. 2019. Trends in the diffusion of misinformation on social media. *Research & Politics*, 6(2):2053168019848554.

Jennifer Allen, Antonio A. Arechar, Gordon Pennycook, and David G. Rand. 2021a. Scaling up fact-checking using the wisdom of crowds. *Science Advances*, 7(36):eabf4393.

Jennifer Allen, Baird Howland, Markus Mobius, David Rothschild, and Duncan J. Watts. 2020. Evaluating the fake news problem at the scale of the information ecosystem. *Science Advances*, 6(14):eaay3539.

Jennifer N L Allen, Cameron Martel, and David G Rand. 2021b. Birds of a feather don't fact-check each other: Partisanship and the evaluation of news in twitter's birdwatch crowdsourced fact-checking program.

Milad Alshomary, Nick Düsterhus, and Henning Wachsmuth. 2020. Extractive snippet generation for arguments. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1969–1972. ACM.

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.

Umme Ammara, Hassan Bukhari, and Junaid Qadir. 2020. Analyzing misinformation through the lens of systems thinking. In *TTO*.

Jesse Dodge an. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *ArXiv preprint*, abs/2002.06305.

Martin Gebser an. 2014. Clingo = ASP + control: Preliminary report. *ArXiv preprint*, abs/1405.3694.

Tianyang Lin An. 2021. A survey of transformers. *ArXiv preprint*, abs/2106.04554.

Martin Armstrong. 2016. Fake news is a real problem.

Phoebe Arnold. 2020. The challenges of online fact checking. Technical report, Full Fact.

Joan S Ash, Marc Berg, and Enrico Coiera. 2004. Some unintended consequences of information technology in health care: the nature of patient care information system-related errors. *Journal of the American Medical Informatics Association*, 11(2):104–112.

Pepa Atanasova, Preslav Nakov, Georgi Karadzhov, Mitra Mohtarami, and Giovanni Da San Martino. 2019. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 1: Check-Worthiness. In *Working Notes of CLEF*, CLEF '19.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020a. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020b. Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364, Online. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735.

Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, and Jakob Grue Simonsen. 2019. MultiFC: A real-world multi-domain dataset for evidence-based fact checking of claims. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4685–4697, Hong Kong, China. Association for Computational Linguistics.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv preprint*, abs/1607.06450.

Mevan Babakar and Will Moy. 2016. The state of automated factchecking.

Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*.

Christopher A. Bail, Lisa P. Argyle, Taylor W. Brown, John P. Bumpus, Haohan Chen, M. B. Fallin Hunzaker, Jaemin Lee, Marcus Mann, Friedolin Merhout, and Alexander Volfovsky. 2010. Exposure to opposing views on social media can increase political polarization. *ArXiv preprint*, abs/10.1073.

Stefano Balietti, Lise Getoor, Daniel G. Goldstein, and Duncan J. Watts. 2010. Reducing opinion polarization: Effects of exposure to similar people with differing political views. *ArXiv preprint*, abs/10.1073.

Meital Balmas. 2014. When fake news becomes real: Combined exposure to multiple news sources and political attitudes of inefficacy, alienation, and cynicism. *Communication Research*, 41(3):430–454.

Rachit Bansal, William Scott Paka, Nidhi, Shubhashis Sengupta, and Tanmoy Chakraborty. 2021. Combining exogenous and endogenous signals with a semi-supervised co-attention network for early detection of covid-19 fake tweets. In *Advances in Knowledge Discovery and Data Mining*, pages 188–200, Cham. Springer International Publishing.

Chitta Baral. 2010. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

David La Barbera, Kevin Roitero, Gianluca Demartini, Stefano Mizzaro, and Damiano Spina. 2020. Crowdsourcing truthfulness: The impact of judgment scale and assessor bias. In *ECIR*, volume 12036 of *Lecture Notes in Computer Science*, pages 207–214. Springer.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 932–938. MIT Press.

Md Momen Bhuiyan, Amy X. Zhang, Connie Moon Sehat, and Tanushree Mitra. 2020. Investigating differences in crowdsourced news credibility assessment: Raters, tasks, and expert criteria. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW2).

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Birdwatch. 2021. Birdwatch data.

Ekaba Bisong. 2019. Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pages 59–64. Apress.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2021. Deep neural networks and tabular data: A survey. *ArXiv preprint*, abs/2110.01889.

Bjarte Botnevik, Eirik Sakariassen, and Vinay Setty. 2020. BRENDA: browser extension for fake news detection. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2117–2120. ACM.

Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7456–7463. AAAI Press.

Julii Brainard and Paul R Hunter. 2020. Misinformation making a disease outbreak worse : outcomes compared for influenza , monkeypox , and norovirus. *Simulation: transactions of the society for modeling, and simulation*, 96(4):365–374.

Erik Brand, Kevin Roitero, Michael Soprano, and Gianluca Demartini. 2021. E-BART: jointly predicting and explaining truthfulness. In *TTO*, pages 18–27.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Tien Duc Cao, Ioana Manolescu, and Xavier Tannier. 2018. Searching for truth in a database of statistics. In *WebDB*, pages 4:1–4:6.

Riccardo Cappuzzo. 2022. *Deep learning models for tabular data curation*. Ph.D. thesis. © EURECOM. Personal use of this material is permitted. The definitive version of this paper was published in Thesis and is available at :.

Carlos Carrasco-Farré. 2022. The fingerprints of misinformation: how deceptive content differs from reliable sources in terms of cognitive effort and appeal to emotions. *Humanities and Social Sciences Communications*, 9(1):162.

Sylvie Cazalens, Philippe Lamarre, Julien Leblay, Ioana Manolescu, and Xavier Tannier. 2018. A content management perspective on fact-checking. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 565–574, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. 2017. Metrics for community analysis: A survey. *ACM Comput. Surv.*, 50(4).

Alessandro Checco, Kevin Roitero, Eddy Maddalena, Stefano Mizzaro, and Gianluca Demartini. 2017. Let's agree to disagree: Fixing agreement measures for crowdsourcing. In *Fifth AAAI Conference on Human Computation and Crowdsourcing*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Lihu Chen, Gael Varoquaux, and Fabian Suchanek. 2022. Imputing out-of-vocabulary embeddings with LOVE makes LanguageModels robust with little cost. In *Proceedings*

of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3488–3504, Dublin, Ireland. Association for Computational Linguistics.

Tongfei Chen, Zhengping Jiang, Adam Poliak, Keisuke Sakaguchi, and Benjamin Van Durme. 2020a. Uncertain natural language inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8772–8779, Online. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020c. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

CHEQ. 2019. The economic cost of bad actors on the internet.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Christos Christodoulopoulos, James Thorne, Andreas Vlachos, Oana Cocarascu, and Arpit Mittal, editors. 2020. *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics, Online.

Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv preprint*, abs/1412.3555.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019b. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Tiphaine Claveau. 2020. CoronaCheck : démêler le vrai du faux sur l'épidémie de covid-19.

Alistair Coleman. 2020. 'hundreds dead' because of covid-19 misinformation.

Keith Coleman. 2021a. Introducing birdwatch, a community-based approach to misinformation.

Keith Coleman. 2021b. Introducing birdwatch, a community-based approach to misinformation.

Alexis Conneau and Guillaume Lample. 2019a. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7057–7067.

Alexis Conneau and Guillaume Lample. 2019b. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7057–7067.

Wanyun Cui and Xingran Chen. 2021. Open rule induction. In *Advances in Neural Information Processing Systems*, volume 34, pages 28536–28547. Curran Associates, Inc.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

D. J. Daley and D. G. Kendall. 1964. Epidemics and rumours. *Nature*, 204(4963):1118–1118.

Fahim Dalvi, Abdul Rafae Khan, Firoj Alam, Nadir Durrani, Jia Xu, and Hassan Sajjad. 2022. Discovering latent concepts learned in BERT. In *International Conference on Learning Representations*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning*

Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Artur S. d'Avila Garcez and L. Lamb. 2020. Neurosymbolic ai: The 3rd wave. *ArXiv preprint*, abs/2012.05876.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: table understanding through representation learning. *Proceedings of the VLDB Endowment*, 14(3):307–319.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Caitlin Dewey. 2016. Facebook fake-news writer: 'i think donald trump is in the white house because of me'. Accessed: 2016-11-17.

Sarthika Dhawan, Siva Charan Reddy Gangireddy, Shiv Kumar, and Tanmoy Chakraborty. 2019. Spotting collective behaviour of online frauds in customer reviews. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 245–251. ijcai.org.

Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2016. Scheduling human intelligence tasks in multi-tenant crowd-powered systems. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 855–865. ACM.

Haoyu Dong, Shijie Liu, Zhouyu Fu, Shi Han, and Dongmei Zhang. 2019. Semantic structure extraction for spreadsheet tables with a multi-task learning architecture. In *Workshop on Document Intelligence at NeurIPS 2019*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610. ACM.

Shiri Dori-Hacohen, David D. Jensen, and James Allan. 2016. Controversy detection in wikipedia using collective classification. In *Proceedings of the 39th International ACM*

SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016, pages 797–800. ACM.

Shiri Dori-Hacohen, Keen Sung, Jengyu Chou, and Julian Lustig-Gonzalez. 2021. *Restoring Healthy Online Discourse by Detecting and Reducing Controversy, Misinformation, and Toxicity Online*, page 2627–2628. Association for Computing Machinery, New York, NY, USA.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Carsten Eickhoff. 2018. Cognitive biases in crowdsourcing. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 162–170. ACM.

Julian Eisenschlos, Maharshi Gor, Thomas Müller, and William Cohen. 2021. MATE: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 213–220, New York, NY, USA. Association for Computing Machinery.

Jeffrey L. Elman. 2010. Finding structure in time. *ArXiv preprint*, abs/10.1207.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

T. Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019. Checkthat! at clef 2019: Automatic identification and verification of claims. In *ECIR*.

Ziv Epstein, Gordon Pennycook, and David G. Rand. 2020. Will the crowd game the algorithm?: Using layperson judgments to combat misinformation on social media by downranking distrusted sources. In *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*, pages 1–11. ACM.

Boi Faltings, Radu Jurca, Pearl Pu, and Bao Duy Tran. 2014. Incentives to counter bias in human computation. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 2(1):59–66.

Angela Fan, Aleksandra Piktus, Fabio Petroni, Guillaume Wenzek, Marzieh Saeidi, Andreas Vlachos, Antoine Bordes, and Sebastian Riedel. 2020a. Generating fact checking briefs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7147–7161, Online. Association for Computational Linguistics.

Angela Fan, Aleksandra Piktus, Fabio Petroni, Guillaume Wenzek, Marzieh Saeidi, Andreas Vlachos, Antoine Bordes, and Sebastian Riedel. 2020b. Generating fact checking briefs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7147–7161, Online. Association for Computational Linguistics.

Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Commun. ACM*, 59(7):96–104.

Daan Fierens, Guy Van den Broeck, Joris Renkens, D. Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. 2014. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, 15:358 – 401.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382.

Kunihiko Fukushima. 1975. Cognitron: A self-organizing multilayer neural network. *Biological Cybernetics*, 20:121–136.

FullFactFAQ. 2022. Full fact frequently asked questions. `https://fullfact.org/about/frequently-asked-questions/`.

Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2019. Exfakt: A framework for explaining facts over knowledge graphs and text. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 87–95. ACM.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, 24(6):707–730.

Siva Charan Reddy Gangireddy, Deepak P, Cheng Long, and Tanmoy Chakraborty. 2020. Unsupervised fake news detection: A graph-based approach. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, HT '20, page 75–83, New York, NY, USA. Association for Computing Machinery.

Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Lisbon, Portugal. Association for Computational Linguistics.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406, Doha, Qatar. Association for Computational Linguistics.

Wolfgang Gatterbauer, Magdalena Balazinska, Nodira Khoussainova, and Dan Suciu. 2009. Believe It or Not: Adding Belief Annotations to Databases. *ArXiv preprint*, abs/0912.5241.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *ArXiv preprint*, abs/2203.14680.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021a. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021b. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *ArXiv preprint*, abs/2106.11959.

Maarten Grootendorst. 2020. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics.

Yue Guo, Yi Yang, and Ahmed Abbasi. 2022a. Auto-debias: Debiasing masked language models with automated biased prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1012–1023, Dublin, Ireland. Association for Computational Linguistics.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022b. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Ashim Gupta and Vivek Srikumar. 2021. X-fact: A new benchmark dataset for multilingual fact checking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 675–682, Online. Association for Computational Linguistics.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2030–2041.

Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Neural check-worthiness ranking with weak supervision: Finding sentences for fact-checking. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 994–1000, New York, NY, USA. Association for Computing Machinery.

Naeemul Hassan, Bill Adair, James T Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The quest to automate fact-checking. In *Proceedings of the 2015 computation+ journalism symposium*.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017a. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1803–1812. ACM.

Naeemul Hassan, Mohammad Yousuf, Md Mahfuzul Haque, Javier A. Suarez Rivas, and Md Khadimul Islam. 2019. Examining the roles of automation, crowds and professionals towards sustainable fact-checking. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 1001–1006, New York, NY, USA. Association for Computing Machinery.

Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, Vikas Sable, Chengkai Li, and Mark Tremayne. 2017b. Claimbuster: The first-ever end-to-end fact-checking system. *Proc. VLDB Endow.*, 10(12):1945–1948.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Chadi Helwe, Chloé Clavel, and Fabian M. Suchanek. 2021. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *3rd Conference on Automated Knowledge Base Construction*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Matthew Hindman and Vlad Barash. 2018. Disinformation, 'fake news'and influence campaigns on twitter.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Angie Drobnic Holan. 2018. The principles of the truth-o-meter. `https://www.politifact.com/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i`.

Jimin Hong, TaeHee Kim, Hyesu Lim, and Jaegul Choo. 2021. AVocaDo: Strategy for adapting vocabulary to downstream domain. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4692–4700, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jeff Horwitz. 2020. Facebook's fact checkers fight surge in fake coronavirus claims.

Arian Hosseini, Siva Reddy, Dzmitry Bahdanau, R Devon Hjelm, Alessandro Sordoni, and Aaron Courville. 2021. Understanding by understanding not: Modeling negation in language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1301–1312, Online. Association for Computational Linguistics.

Mohammad Hosseini, Medard Hilhorst, Inez de Beaufort, and Daniele Fanelli. 2018. Doing the right thing: A qualitative investigation of retractions due to unintentional error. *Science and engineering ethics*, 24(1):189–206.

Lee Howell. 2013. Global risks 2013.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420, Berlin, Germany. Association for Computational Linguistics.

Christoph Hube, Besnik Fetahu, and Ujwal Gadiraju. 2019. Understanding and mitigating worker biases in the crowdsourced collection of subjective judgments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*, page 407. ACM.

Larry Huynh, Thai Nguyen, Joshua Goh, Hyoungshick Kim, and Jin B Hong. 2021. Argh! automated rumor generation hub. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3847–3856.

Viet-Phi Huynh and Paolo Papotti. 2019. Buckle: Evaluating fact checking algorithms built on knowledge bases. *PVLDB*, 12(12):1798–1801.

Yusra Ibrahim, Mirek Riedewald, Gerhard Weikum, and Demetrios Zeinalipour-Yazti. 2019. Bridging quantities in tables and text. In *ICDE*, pages 1010–1021.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. 2014. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2492–2500.

Maurice Jakesch, Moran Koren, Anna Evtushenko, and Mor Naaman. 2020. How partisan crowds affect news evaluation. In *TTO*.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In

Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pages 26–30, New Orleans, Louisiana. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Ralph Jennings. 2020. Why vietnam is asking other asian countries to help squelch fake news.

Shaoxiong Ji, Shirui Pan, E. Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33:494–514.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Saehan Jo, Immanuel Trummer, Weicheng Yu, Xuezhi Wang, Cong Yu, Daniel Liu, and Niyati Mehta. 2019. Verifying text summaries of relational data sets. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 299–316. ACM.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Daniel Kahneman, Yann LeCun, Yoshua Bengio, and Geoffery Hinton. 2020. Fireside chat with daniel kahneman. ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE.

Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. CoronaCheck.

Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A Mixed-Initiative Approach to Large-Scale, Data-Driven Claim Verification (Technical Report).

Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: Fact checking statistical claims. VLDB 2020, 46th International Conference on Very Large Data Bases, 31 August-4 September 2020, Tokyo, Japan (Virtual Conference) / To be published in PVLDB (Proceedings of the VLDB Endowment), Vol.13, N°12, 2020.

Georgios Karagiannis, Immanuel Trummer, Saehan Jo, Shubham Khandelwal, Xuezhi Wang, and Cong Yu. 2019. Mining an “anti-knowledge base" from wikipedia updates with applications to fact checking and beyond. *PVLDB*, 13(4):561–573.

Nora Kassner, Benno Krojer, and Hinrich Schütze. 2020. Are pretrained language models symbolic reasoners over knowledge? In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 552–564, Online. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Gabriella Kazai. 2011. In search of quality in crowdsourcing for search engine evaluation. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval - Volume 6611*, ECIR 2011, page 165–176, Berlin, Heidelberg. Springer-Verlag.

Ashkan Kazemi, Kiran Garimella, Gautam Kishore Shahi, Devin Gaffney, and Scott A. Hale. 2021. Tiplines to combat misinformation on encrypted platforms: A case study of the 2019 indian election on whatsapp. *HKS Misinfo Review Journal*, abs/2106.04726.

Guolin Ke, Di He, and Tie-Yan Liu. 2021a. Rethinking positional encoding in language pre-training. In *Proceedings of the 9th International Conference on Learning Representations*, ICLR ’21.

Zixuan Ke, Hu Xu, and Bing Liu. 2021b. Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4746–4755, Online. Association for Computational Linguistics.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3390–3398. AAAI Press.

Glenn Kessler. 2017. About the fact checker. `https://www.washingtonpost.com/politics/2019/01/07/about-fact-checker`.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2021. A distributional approach to controlled text generation. In *International Conference on Learning Representations*.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative

testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682. PMLR.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Alksi Knuttila, Lisa-Maria Neudert, and Philip N. Howard. 2022. Who is afraid of fake news? modeling risk perceptions of misinformation in 142 countries.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2018. Towards automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. *ArXiv preprint*, abs/1809.08193.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2017. Learning active learning from data. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4225–4235.

Neema Kotonya and Francesca Toni. 2020. Explainable automated fact-checking: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5430–5443, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Katie Langin. 2018. Fake news spreads faster than true news on twitter—thanks to people, not bots.

Latvian Public Broadcasting. 2020. 130 countries sign up for infodemic pledge.

Patrick R. Laughlin, Megan L Zander, Erica M Knievel, and Tiong Kean Tan. 2003. Groups perform better than the best individuals on letters-to-numbers problems: informative equations and effective strategies. *Journal of personality and social psychology*, 85 4:684–94.

Anne Lauscher, Tobias Lueken, and Goran Glavaš. 2021. Sustainable modular debiasing of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4782–4797, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Julien Leblay. 2017. A declarative approach to data-driven fact checking. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 147–153. AAAI Press.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Joohyung Lee, Samidh Talsania, and Y. Wang. 2017. Computing LPMLN using ASP and MLN solvers. *Theory and Practice of Logic Programming*, 17(5–6):942–960.

Joohyung Lee and Yi Wang. 2016. Weighted rules under the stable model semantics. In *KR*, pages 145–154.

Nayeon Lee, Belinda Z. Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. 2020. Language models as fact checkers? In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 36–41, Online. Association for Computational Linguistics.

Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. 2021a. How is BERT surprised? layerwise detection of linguistic anomalies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4215–4228, Online. Association for Computational Linguistics.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *PVLDB*, 8(1):73–84.

Fei Li and HV Jagadish. 2016. Understanding natural language queries over relational databases. *SIGMOD Record*, 45(1):6–13.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021b. Pretrained language model for text generation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4492–4499. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Tao Li and Vivek Srikumar. 2019. Augmenting neural networks with first-order logic. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 292–302, Florence, Italy. Association for Computational Linguistics.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16.

Percy Liang. 2016. Learning Executable Semantic Parsers for Natural Language Understanding. *Commun. ACM*, 59(9):68–76.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Di Liu, Randolph G. Bias, Matthew Lease, and Rebecca Kuipers. 2010. Crowdsourcing for usability testing. *ArXiv preprint*, abs/10.1002.

Haochen Liu, Joseph Thekinen, Sinem Mollaoglu, Da Tang, Ji Yang, Youlong Cheng, Hui Liu, and Jiliang Tang. 2022a. Toward annotator group bias in crowdsourcing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1797–1806, Dublin, Ireland. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020a. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022b. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

Yang Liu and Yi-fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 354–361. AAAI Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Mansoureh Maadi, Hadi Akbarzadeh Khorshidi, and Uwe Aickelin. 2021. A review on human–ai interaction in machine learning and insights for medical applications. *International Journal of Environmental Research and Public Health*, 18.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

Christopher Malon. 2021. Team papelo at FEVEROUS: Multi-hop evidence pursuit. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 40–49, Dominican Republic. Association for Computational Linguistics.

Jonathan Mamou, Hang Le, Miguel Del Rio, Cory Stephenson, Hanlin Tang, Yoon Kim, and SueYeon Chung. 2020. Emergence of separable manifolds in deep language representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6713–6723. PMLR.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3753–3763.

Alexios Mantzarlis. 2015. Will verification kill fact-checking?

Alexios Mantzarlis. 2018. Module 5: Fact-checking 101.

Ben McCamish, Vahid Ghadakchi, Arash Termehchy, Liang Huang, and Behrouz Touri. 2019. How do humans and data systems establish a common query language? *SIGMOD Record*, 48(1):51–58.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Johannes Mario Meissner, Napat Thumwanit, Saku Sugawara, and Akiko Aizawa. 2021. Embracing ambiguity: Shifting the training target of NLI models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 862–869, Online. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual knowledge in gpt. *ArXiv preprint*, abs/2202.05262.

Martino Mensio and Harith Alani. 2019. Misinfome: Who is interacting with misinformation? In *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019*, volume 2456 of *CEUR Workshop Proceedings*, pages 217–220. CEUR-WS.org.

Meta. 2016. Meta's third-party fact-checking program.

Meta. 2021. Third party fact-checking. https://www.facebook.com/journalismproject/programs/third-party-fact-checking/how-it-works.

Nicholas Micallef, Bing He, Srijan Kumar, Mustaque Ahamad, and Nasir Memon. 2020. The role of the crowd in countering misinformation: A case study of the covid-19 infodemic. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 748–757.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5309–5316. AAAI Press.

Barbara Milucci. 2020. Tutelare le persone dal rischio infodemia: nasce un sito contro le fake news.

Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. 2020. Differentiable reasoning on large knowledge bases and natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5182–5190.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *Proceedings*

of the 9th International Conference on Learning Representations, ICLR '21, Virtual Event, Austria. OpenReview.net.

Preslav Nakov, David Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021a. Automated fact-checking for assisting human fact-checkers. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4551–4558. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Preslav Nakov, Giovanni Da San Martino, Tamer Elsayed, Alberto Barrón-Cgedeño, Rubén Míguez, Shaden Shaar, Firoj Alam, Fatima Haouari, Maram Hasanain, Nikolay Babulkov, Alex Nikolov, Gautam Kishore Shahi, Julia Maria Struß, and Thomas Mandl. 2021b. The clef-2021 checkthat! lab on detecting check-worthy claims, previously fact-checked claims, and fake news. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 639–649, Berlin, Heidelberg. Springer-Verlag.

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *ArXiv preprint*, abs/2205.09911.

GL Nemhauser and LA Wolsey. 1978. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188.

Dorit Nevo and Benjamin D. Horne. 2022. How topic novelty impacts the effectiveness of news veracity interventions. *Commun. ACM*, 65(2):68–75.

An T. Nguyen, Aditya Kharosekar, Saumyaa Krishnan, Siddhesh Krishnan, Elizabeth Tate, Byron C. Wallace, and Matthew Lease. 2018. Believe it or not: Designing a human-ai partnership for mixed-initiative fact-checking. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 189–199, New York, NY, USA. Association for Computing Machinery.

NllbTeam, Marta Ruiz Costa-jussà, James Cross, Onur cCelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Alison Youngblood, Bapi Akula, Loïc Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon L. Spruit, C. Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzm'an, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *ArXiv preprint*, abs/2207.04672.

Uyiosa Omoregie. 2021. Misinformation analysis and online quality theory (a wittgensteinian approach).

World Health Organization. 2022. Covid-19 research and innovation powering the world's pandemic response – now and in the future.

Stefano Ortona, Vamsi Meduri, and Paolo Papotti. 2018. Robust discovery of positive and negative rules in knowledge-bases. In *ICDE*. http://www.eurecom.fr/fr/publication/5321.

Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. 2018a. Robust Discovery of Positive and Negative Rules in Knowledge Bases. In *IEEEICDE*, pages 1168–1179.

Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. 2018b. Rudik: Rule discovery in knowledge bases. *Proc. VLDB Endow.*, 11(12):1946–1949.

Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2021. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):604–624.

Nedjma Ousidhoum, Xinran Zhao, Tianqing Fang, Yangqiu Song, and Dit-Yan Yeung. 2021. Probing toxic content in large pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4262–4274, Online. Association for Computational Linguistics.

Hannah Jane Parkinson. 2016. Click and elect: how fake news helped donald trump win a real election. Accessed: 2016-11-14.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3973–3997, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694.

David J Pearce and Paul HJ Kelly. 2007. A dynamic topological sort algorithm for directed acyclic graphs. *Journal of Experimental Algorithmics (JEA)*, 11:1–7.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Marcos Rodrigues Pinto, Yuri Oliveira de Lima, Carlos Eduardo Barbosa, and Jano Moreira de Souza. 2019. Towards fact-checking through crowdsourcing. In *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 494–499.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Julie Posetti and Alice Matthews. 2018. A short guide to the history of 'fake news' and disinformation: A new icfj learning module.

Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. 2021. *Vera: Prediction Techniques for Reducing Harmful Misinformation in Consumer Health Search*, page 2066–2070. Association for Computing Machinery, New York, NY, USA.

Nicolas Pröllochs. 2021. Community-based fact-checking on twitter's birdwatch platform. *ArXiv preprint*, abs/2104.07175.

Giovanni Puccetti, Alessio Miaschi, and Felice Dell'Orletta. 2021. How do BERT embeddings organize linguistic knowledge? In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 48–57, Online. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *ArXiv preprint*, abs/2003.08271.

Yunke Qu, David La Barbera, Kevin Roitero, Stefano Mizzaro, Damiano Spina, and Gianluca Demartini. 2022. Combining human and machine confidence in truthfulness assessment. *J. Data and Information Quality*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Mario Rautner. 2020. Evaluating evidence and information sources. `https://kit.exposingtheinvisible.org/en/how/evaluate-evidence.html`.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ehud Reiter and Craig Thomson. 2020. Shared task on evaluating accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 227–231, Dublin, Ireland. Association for Computational Linguistics.

Paul Resnick, Aljohara Alfayez, Jane Im, and Eric Gilbert. 2021. Informed crowds can effectively identify misinformation. *ArXiv preprint*, abs/2108.07898.

Rayhane Rezgui, Mohammed Saeed, and Paolo Papotti. 2021. Automatic verification of data summaries. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 271–275, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *TREC*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Kevin Roitero, Michael Soprano, Shaoyang Fan, Damiano Spina, Stefano Mizzaro, and Gianluca Demartini. 2020a. Can the crowd identify misinformation objectively?: The effects of judgment scale and assessor's background. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 439–448. ACM.

Kevin Roitero, Michael Soprano, Beatrice Portelli, Massimiliano De Luise, Damiano Spina, Vincenzo Della Mea, Giuseppe Serra, Stefano Mizzaro, and Gianluca Demartini. 2021. Can the crowd judge truthfulness? a longitudinal study on recent misinformation about covid-19. *Personal and Ubiquitous Computing*, pages 1 – 31.

Kevin Roitero, Michael Soprano, Beatrice Portelli, Damiano Spina, Vincenzo Della Mea, Giuseppe Serra, Stefano Mizzaro, and Gianluca Demartini. 2020b. The COVID-19 infodemic: Can the crowd judge recent misinformation objectively? In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 1305–1314. ACM.

Uma Roy, Noah Constant, Rami Al-Rfou, Aditya Barua, Aaron Phillips, and Yinfei Yang. 2020. LAReQA: Language-agnostic answer retrieval from a multilingual pool. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5919–5930, Online. Association for Computational Linguistics.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, San Diego, California. Association for Computational Linguistics.

Stuart Russell and Peter Norvig. 2010. *Artificial Intelligence: A Modern Approach*, 3 edition. Prentice Hall.

Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. 2021a. RuleBERT: Teaching soft rules to pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1460–1476, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mohammed Saeed, Giulio Alfarano, Khai Nguyen, Duc Pham, Raphael Troncy, and Paolo Papotti. 2021b. Neural re-rankers for evidence retrieval in the FEVEROUS task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 108–112, Dominican Republic. Association for Computational Linguistics.

Mohammed Saeed and Paolo Papotti. 2021. Fact-checking statistical claims with tables. IEEE Data Engineering Bulletin, August 2021.

Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2018. Using link features for entity clustering in knowledge graphs. In *European Semantic Web Conference*, pages 576–592. Springer.

Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Ozcan. 2016. ATHENA: An ontology-driven system for natural language querying over relational data stores. *VLDB*, 9(12):1209–1220.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Soumya Sanyal, Harman Singh, and Xiang Ren. 2022. FaiRR: Faithful and robust deductive reasoning over natural language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1075–1093, Dublin, Ireland. Association for Computational Linguistics.

Jessica Schrouff, Sebastien Baur, Shaobo Hou, Diana Mincu, Eric Loreaux, Ralph Blanes, James Wexler, Alan Karthikesalingam, and Been Kim. 2021. Best of both worlds: local and global explanations with human-understandable concepts. *ArXiv preprint*, abs/2106.08641.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shaden Shaar, Nikolay Babulkov, Giovanni Da San Martino, and Preslav Nakov. 2020. That is a known lie: Detecting previously fact-checked claims. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3607–3618, Online. Association for Computational Linguistics.

Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. 2019. Combating fake news: A survey on identification and mitigation techniques. *ACM Trans. Intell. Syst. Technol.*, 10(3).

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Elisa Shearer. 2021. More than eight-in-ten americans get news from digital devices.

Wanita Sherchan, Surya Nepal, and Cecile Paris. 2013. A survey of trust in social networks. *ACM Comput. Surv.*, 45(4):47:1–47:33.

Baoxu Shi and Tim Weninger. 2016. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-Based Systems*, 104:123–133.

Jamin Shin, Andrea Madotto, and Pascale Fung. 2018. Interpreting word embeddings with eigenvector analysis. In *Workshop on Interpretability and Robustness in Audio, Speech, and Language (IRASL)*. NeurIPS IRASL.

Jieun Shin, Lian Jian, Kevin Driscoll, and François Bar. 2017. Political rumoring on twitter during the 2012 us presidential election: Rumor diffusion and correction. *New Media & Society*, 19(8):1214–1235.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Arnab Shome. 2017. John mcafee twitter handle hack results in pump and dump of multiple coins.

Kai Shu, H. Russell Bernard, and Huan Liu. 2018. Studying fake news via network analysis: Detection and mitigation. *ArXiv preprint*, abs/1804.10233.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36.

Shivangi Singhal, Rajiv Ratn Shah, Tanmoy Chakraborty, Ponnurangam Kumaraguru, and Shin'ichi Satoh. 2019. Spotfake: A multi-modal framework for fake news detection. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 39–47.

Antonio Spadaro. 2020. CoronaCheck and FakeNews.

Robyn Speer. 2017. Conceptnet numberbatch 17.04: better, less-stereotyped word vectors. `http://blog.conceptnet.io/posts/2017/conceptnet-numberbatch-17-04-better-less-stereotyped-word-vectors/`.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey. European Language Resources Association (ELRA).

Dan Sperber and Hugo Mercier. 2018. *The Enigma of Reason*. Harvard University Press, Cambridge, MA and London, England.

Stanford History Education Group. 2016. *Evaluating Information: The Cornerstone of Civic Online Reasoning : Executive Summary*. Stanford University.

Kate Starbird. 2019. Disinformation's spread: bots, trolls and all of us. *Nature*, 571(7766):449–450.

Mark Stencel, Erica Ryan, and Joel Luther. 2022. Fact-checkers extend their global reach with 391 outlets, but growth has slowed.

D.P. Stroh. 2015. *Systems Thinking For Social Change: A Practical Guide to Solving Complex Problems, Avoiding Unintended Consequences, and Achieving Lasting Results*. Chelsea Green Publishing.

Ting Su, Craig Macdonald, and Iadh Ounis. 2019. Ensembles of recurrent networks for classifying the relationship of fake news titles. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 893–896. ACM.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706. ACM.

Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. *Annotating Columns with Pre-trained Language Models*. Association for Computing Machinery.

Megha Sundriyal, Ganeshan Malhotra, Md Shad Akhtar, Shubhashis Sengupta, Andrew Fano, and Tanmoy Chakraborty. 2022. Document retrieval and claim verification to mitigate COVID-19 misinformation. In *Proceedings of the Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situations*, pages 66–74, Dublin, Ireland. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020a. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020b. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33 of *NeurIPS '20*, pages 20227–20237, Online.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Comput. Surv.* Just Accepted.

Orkun Temiz, Özgün Ozan Kılıç, Arif Ozan Kızıldağ, and Tuğba Taşkaya Temizel. 2021. A fact checking and verification system for FEVEROUS using a zero-shot learning approach. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 113–120, Dominican Republic. Association for Computational Linguistics.

Saravanan Thirumuruganathan, Nan Tang, Mourad Ouzzani, and AnHai Doan. 2020. Data curation with deep learning. In *EDBT*.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. Database reasoning over text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3091–3104, Online. Association for Computational Linguistics.

Sebastian Tschiatschek, Adish Kumar Singla, Manuel Gomez-Rodriguez, Arpit Merchant, and Andreas Krause. 2017. Detecting fake news in social networks via crowdsourcing. *ArXiv preprint*, abs/1711.09025.

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022a. Investigating selective prediction approaches across several tasks in IID, OOD, and adversarial settings. In

Findings of the Association for Computational Linguistics: ACL 2022, pages 1995–2002, Dublin, Ireland. Association for Computational Linguistics.

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022b. Towards improving selective prediction ability of NLP systems. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 221–226, Dublin, Ireland. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Enzo Veltri, Donatello Santoro, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2022. Pythia: Unsupervised generation of ambiguous textual claims from relational data. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, page 2409–2412, New York, NY, USA. Association for Computing Machinery.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Andreas Vlachos. 2008. A stopping criterion for active learning. *Computer Speech & Language*, 22(3):295–312.

Andreas Vlachos. 2020. Fact-checking as a conversation. https://www.ims.uni-stuttgart.de/documents/kolloquium/Andreas-Vlachos_Fact-Checking-as-a-Conversation-Spring-2022.pdf.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA. Association for Computational Linguistics.

Nguyen Vo and Kyumin Lee. 2018. The rise of guardians: Fact-checking URL recommendation to combat fake news. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 275–284. ACM.

Nguyen Vo and Kyumin Lee. 2020. Where are the facts? searching for fact-checked information to alleviate the spread of fake news. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7717–7731, Online. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3774–3785, Hong Kong, China. Association for Computational Linguistics.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021a. On position embeddings in {bert}. In *International Conference on Learning Representations*.

Dong Wang, Lance Kaplan, Hieu Le, and Tarek Abdelzaher. 2012. On truth discovery in social sensing: A maximum likelihood estimation approach. In *2012 ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 233–244.

Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro Szekely. 2021b. Retrieving complex tables with multi-granular graph representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1472–1482.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021c. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Xuezhi Wang, Cong Yu, Simon Baumgartner, and Flip Korn. 2018b. Relevant document discovery for fact-checking articles. In *The Web Conf.*, pages 525–533.

Yu-An Wang and Yun-Nung Chen. 2020. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, Online. Association for Computational Linguistics.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021d. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Fabian L. Wauthier and Michael I. Jordan. 2011. Bayesian bias mitigation for crowdsourcing. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 1800–1808.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. Association for Computational Linguistics.

Penghui Wei, Nan Xu, and Wenji Mao. 2019. Modeling conversation structure and temporal dynamics for jointly predicting rumor stance and veracity. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4787–4798, Hong Kong, China. Association for Computational Linguistics.

Nathaniel Weir, Andrew Crotty, Alex Galakatos, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Ugur Cetintemel, Prasetya Utama, Nadja Geisler, Benjamin Hättasch,

Steffen Eger, and Carsten Binnig. 2019. DBPal: Weak Supervision for Learning a Natural Language Interface to Databases. *ArXiv preprint*, abs/1909.06182.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Wikipedia. 2020. Misinformation related to the COVID-19 pandemic.

Dustin Wright and Isabelle Augenstein. 2020. Claim check-worthiness detection as positive unlabelled learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 476–488, Online. Association for Computational Linguistics.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. 2022. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*.

Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv preprint*, abs/1609.08144.

You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward computational fact-checking. *Proceedings of the VLDB Endowment*, 7(7):589–600.

Yaqi Xie, Ziwei Xu, Kuldeep S. Meel, Mohan S. Kankanhalli, and Harold Soh. 2019. Embedding symbolic knowledge into deep networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4235–4245.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Peng Xu, Xiatian Zhu, and David A. Clifton. 2022. Multimodal learning with transformers: A survey. *ArXiv preprint*, abs/2206.06488.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2319–2328.

Ziyi Yang, Yinfei Yang, Daniel Cer, and Eric Darve. 2021. A simple and effective method to eliminate the self language bias in multilingual representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5825–5832, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *ArXiv preprint*, abs/2009.13845.

Hanlin Zhang, Ziyang Li, Jiani Huang, Mayur Naik, and Eric Xing. 2022. Improved logical reasoning of language models via differentiable symbolic programming. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*.

Huaiwen Zhang, Quan Fang, Shengsheng Qian, and Changsheng Xu. 2019. Multi-modal knowledge-aware event memory network for social media rumor detection. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, pages 1942–1951.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *ArXiv preprint*, abs/1709.00103.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

Xiang Zhou, Heba Elfardy, Christos Christodoulopoulos, Thomas Butler, and Mohit Bansal. 2021. Hidden biases in unreliable news detection datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2482–2492, Online. Association for Computational Linguistics.

Xinyi Zhou, Jindi Wu, and Reza Zafarani. 2020a. Safe: Similarity-aware multi-modal fake news detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 354–367. Springer.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020b. Evaluating commonsense in pre-trained language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9733–9740.

A Additional Material For Chapter 3

A.1 CoronaCheck Data Generation

We did not have any initial dataset to bootstrap CoronaCheck. For this, we used values from the relations themselves to generate the training sentences for the classifiers. For example, we have a table for the number of deaths in each country and across several days and months. By using simple templates and queries, we generate sentences such as "*There are* **82279** *cases in* **China** *in* **March**", where the bold tokens are coming from the query result and the italic ones are coming from the template. The query results cover different country names and dates, leading to numerous possible sentences. We follow the same process across the different tables with the number of deaths, recovered cases and so on.

Following the idea in data augmentation, we add variety to the sentences by replacing some keywords with their synonyms. We also add interchange clauses in a sentence. For example, the previously mentioned sentences becomes "In March, there are 82279 cases in China". We generate training sentences in English, French, and Italian. We did not generate data for German, as we found the quality of the multilingual transformer good enough to handle new languages.

We remark that these sentences are not labeled true or false, as we are using them to train the classifiers (e.g., recognize the right relation, formula, etc.), they are not training examples for the verification step.

A.2 Claim Preprocessing

For mapping a claim to the options in the query spaces, we rely on classifiers built on top of pre-trained embeddings, which map each word to a real-valued vector (Pennington et al., 2014; Conneau and Lample, 2019b). Our solution is independent of the underlying classifier (as long as it can express a confidence measure) and of the pre-trained embeddings.

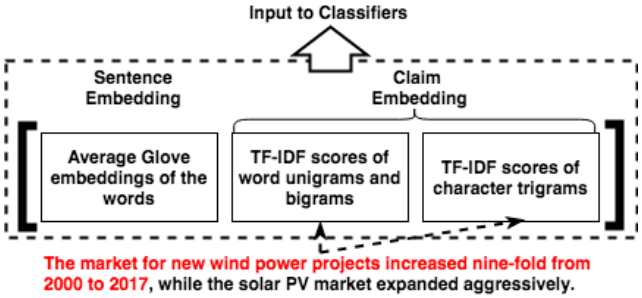

Figure A.1: Preprocessing of the claims.

To get the embedding of a sentence, we average the embedding of each word in that sentence. As a sentence may contain more than one claim, we enrich the representation with claim-specific features. As depicted in Figure A.1, for every claim, we concatenate the sentence embedding with the TF-IDF scores of the claim unigrams and bigrams, followed by the TF-IDF scores of every 3 characters. N-grams enable the learning of sub strings in the claim, such as "**non**-increasing". If the claim is explicit, we identify the value directly from the sentence with a syntactical parsing.

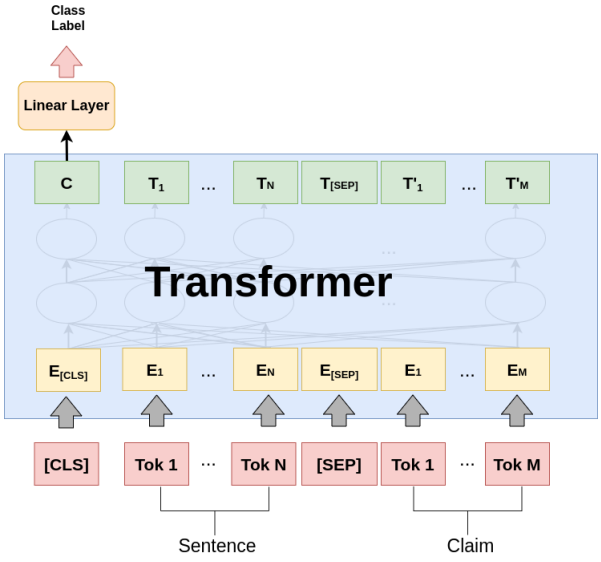

Figure A.2: Transformer Input and Architecture

When dealing with transformers, we input both the sentence and claim as input by first tokenizing them, and then use the *[SEP]* token to join them. The joined tokenized input is fed to the transformer with a linear layer on top of the pooled output to allow for multi-class classification, as shown in Figure A.2.

A.3 More Experimental Results

Table A.1: Percentiles of property value frequencies.

Percentiles	10%	25%	50%	95%	99%
Relation	2	4	10	199	532
Row Index	2	2	4	39	107
Column	1	2	7	127	1400
Formula	1	1	1	8	55

Table A.1 reports percentiles of property value frequencies for $\mathbf{IEA_L}$.

A.3.1 More Results on Error Injection

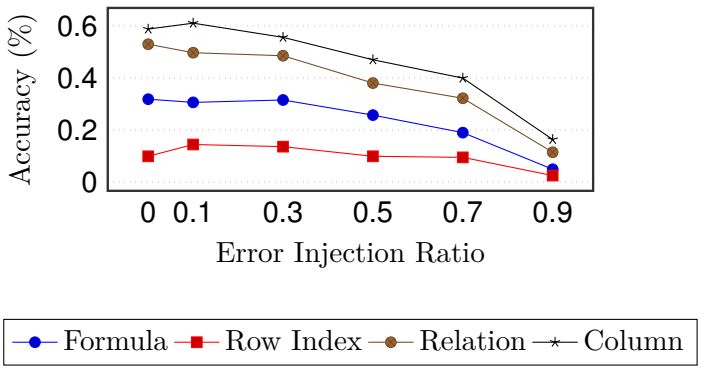

Figure A.3: Variation of classifier top-1 accuracy as a function of ratio of erroneous claims .

Figure A.3 reports the quality results for $k = 1$ with increasing amount of mislabeled training data given as input. Experimental setup is the same as that in Figure 3.10.

A.3.2 Supported Functions

Table A.2: Percentage of claims $\mathbf{IEA_L}$ w.r.t. the functions covered by the baseline methods.

	%
Lookup	11.29%
SUM	0.98%
COUNT	0.06%
AVG	1.05%
MAX	0.3%
ArgMax	0.37%
RANK	1.11%
Comparison	9.13%

We report in Table A.2 a break-down of the percentage of claims that can be covered by the baselines in the $\mathbf{IEA_L}$ dataset.

A.3.3 Variation of Training Set Size

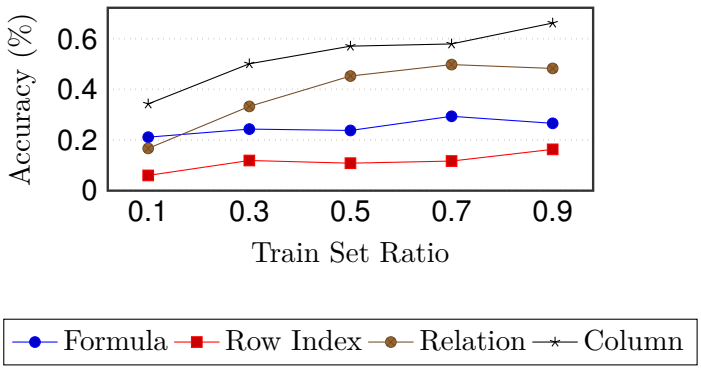

Figure A.4: Variation of classifier top-1 accuracy as a function of training dataset size.

Figure A.4 reports the quality results for $k = 1$ and increasing amount of training data given as input. Experimental setup is the same as that in Figure 3.11.

A.3.4 Freshness vs. Training Data Size

The semantics of certain text fragments changes over time. For instance, a reference to the "current month" changes semantics as the month changes. Training our classifiers

with outdated samples (user input assigning claims to query properties) may lead to suboptimal results. For our CoronaCheck Website, using user-provided training data, we noticed low accuracy for the classifier mapping claims to relevant time periods. We identified training with outdated training data as a likely cause. To verify that hypothesis, we tried training a specialized classifier for each month, using only crowd input obtained during the corresponding month. Note that the total amount of training data used per classifier is lower than before. Hence, we have a tradeoff between the amount and freshness of training data.

Table A.3: Accuracy on $\mathbf{C19_L}$ test data with column classifiers trained on monthly answers from Web users.

Month	Size of Training Set	Accuracy
February	96	0.33
March	920	0.12
April	347	0.49
May	35	0.49
June	18	-
July	7	-

Table 3.6 shows the experimental results. We report the number of training samples obtained from users during a certain month. Also, we report accuracy of the time period classifier (time periods are represented as columns in our database). For each month, we report accuracy on the claims from $\mathbf{C19_L}$, adapting labels assuming that the claim was issued in the corresponding month (e.g., we update labels for claims referring to "the current month"). For June and July, we received a few training samples that all assign the same (current) month as label. We do not report accuracy numbers, as our classifier implementation throws an error if only one label is defined for classification. Accuracy does not necessarily correlate with the number of training samples. For instance, we obtain relatively low accuracy for March despite receiving the highest number of samples. Here, we observed repeated occurrences of low-quality labels obtained from the crowd. On the other side, accuracy is relatively high for April and May. This is explained by the fact that learning to map input claims to the current month as default is most crucial to increase accuracy. This can be achieved using a low number of training samples with high-quality labels. Altogether, using smaller sets of fresh training samples increases accuracy, compared to using all training data (accuracy of 0.04).

A.3.5 Outlier Claim Experiments

Our templates support the most popular claim types. Nevertheless, we occasionally encounter claims that are verified by queries that do not match any of our templates. To

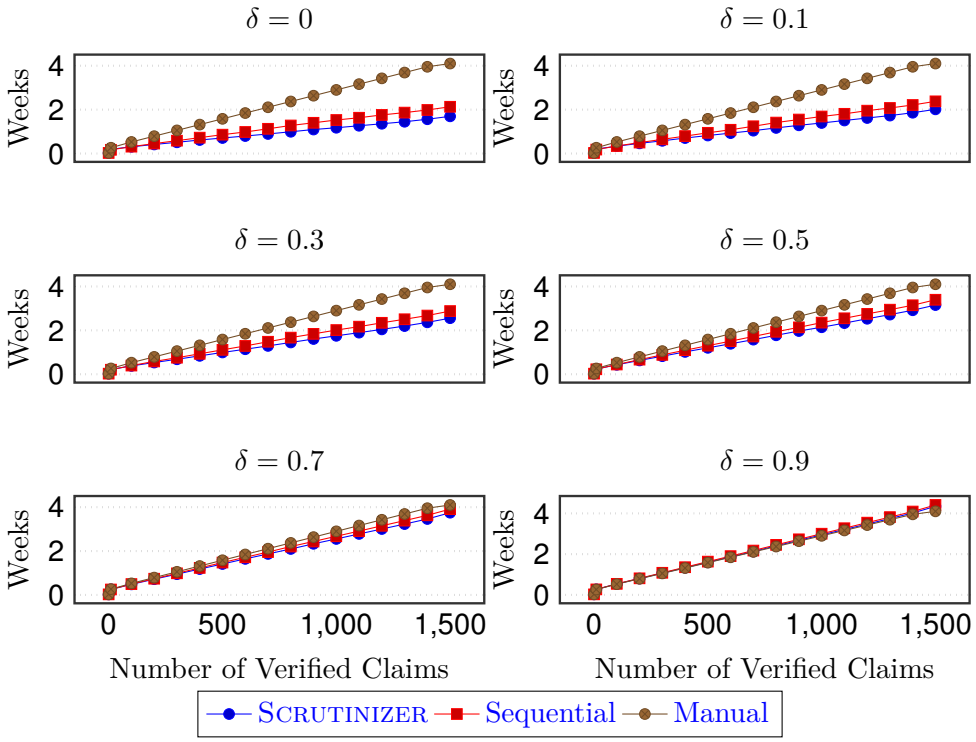

Figure A.5: Simulated verification time versus quality of classifiers.

detect outlier claims, we could, for instance, prompt administrator users to consider an expansion of the current templates if outliers are detected. As a first step, we verified whether claims with outlier queries could be recognized based on classifier confidence. We hypothesize that such claims may correlate with low classifier confidence.

To verify our hypothesis, we collected claims from the CoronaCheck log that cannot be verified with the current templates. We identified ten claims that translate into queries with a `MAX` aggregation function. This case is rather rare, and we do not currently support it. For comparison, we sampled ten other claims from our log (with uniform random distribution) that can be verified using our templates. The average classifier confidence was 0.39 (standard deviation of 0.08) for the outlier claims. It was 0.94 (standard deviation of 0.035) for the randomly sampled claims. Hence, our hypothesis holds in this experiment, motivating outlier detection mechanisms based on classifier confidence.

A.3.6 Simulation with Low-Accuracy Classifiers

The quality of the classifiers may influence the benefit we obtain via techniques such as claim prioritization. We tested that hypothesis by simulating classifiers of varying accuracy in the following experiment (classifier accuracy depends for instance on the quality of training data). We use the same experimental setup as for the results shown in

```
<Query>::="Select" <E>
<E>::=<E>"+"<E>|<E>"-"<E>|<E>"*"<E>|<E>"/"<E>|
  <E>"%"<E>|"("<E>")"|<Const>|<SQ>
<SQ>::="Select" <YR> "From" <TB> "Where" <CD>
<YR>::="2000"|"2001"|...
<TB>::="GlobalCO2"|"GlobalEnergyDemand"|...
<CD>::="Index=" <PV>
<PV>::="PGElecDemand"|"PGINCoal"|...
```

Figure A.6: EBNF representation of query space considered for verifying IEA claims.

Figure 3.6. However, we "disturb" classifiers to varying degrees. We add a disturbance by artificially decreasing the rank of ground truth labels in the classification result. We do so with a disturbance probability δ, varying δ from 0 to 0.9 (i.e., from none to high degree of disturbance).

Figure A.5 reports the results. We find that classifier accuracy influences indeed the performance of Scrutinizer and the sequential baseline. Simulated verification time increases as the accuracy of classification drops. This is due to the fact that correct answers are shown less frequently as answer options. In that case, fact-checkers need to enter the correct option manually, which takes more (simulated) time. Regardless of classifier accuracy, we observe that claim prioritization remains helpful (i.e., Scrutinizer performs better than the Sequential baseline), even if the absolute distance decreases for higher degrees of disturbance. In most cases, Scrutinizer dominates manual verification (simulating the approach taken by the IEA fact-checkers currently). This changes only for very high degrees of disturbance. If answer options suggested by the system are almost certainly incorrect, entering the answer manually is preferable.

A random ordering of the claims has been also experimented with, showing improvements over the sequential baseline, probably since such ordering minimizes uncertainty, but on the expense of more verification cost. Scrutinizer still performs better.

A.4 Query Space for Use Cases

Our system is motivated by two use cases: verifying claims in IEA reports and on the Coronavirus ("CoronaCheck"). We discuss those use cases in detail in the introduction. Due to space constraints, we omitted a full description of the space of queries considered for verification in those two scenarios. We do so in the following. We represent the space of considered queries for each scenario as a grammar, described in Extended Backus-Naur form (EBNF). Alternatively, we could describe those search spaces as query templates (described in Section 3.4). We choose EBNF over query templates due to its conciseness when describing CoronaCheck queries (while the template formalism is more convenient for describing our pseudocode).

Figure A.6 describes the space of queries considered for verifying IEA's queries. It

```
<Query>::="Select" <Bool>
<Bool>::=<C> <Cmp> <C>|<C> <Cmp> <Const>|
  <D> <Cmp> <D>|<D> <Cmp> <Const>|
  <R> <Cmp> <R>|<R> <Cmp> <Const>|
  <B> <Cmp> <B>|<B> <Cmp> <Const>
<cmp>::="<"|">"|"="
<C>::="Select" <Time> "From NrCases Where" <Pred>
<D>::="Select" <Time> "From NrDeaths Where" <Pred>
<R>::="Select" <Time> "From NrDeaths Where" <Pred>
    "/ Select" <Time> "From NrCases Where" <Pred>
<B>::="Select" <Time> "From NrRecov Where" <Pred>
<Pred>::="Cname=" <Country>
```

Figure A.7: EBNF representation of query space considered for verifying claims in CoronaCheck.

corresponds to the query template specified in Example 3. We consider queries that calculate arithmetic expressions, using (numerical) constants (`<Const>`) or lookup sub-queries (`<SQ>`) as operands. The lookup sub-queries are formed by selecting a table, a column representing the year, and an equality condition on the primary key column.

Figure A.7 shows the grammar describing some queries considered for CoronaCheck (as of June 2020, we keep extending the range of supported queries). Compared to IEA, we consider a more narrow space of formulas. Unlike in the case of IEA, the associated query template does not use an EXP element modeling arbitrary arithmetic expressions. For CoronaCheck, we restrict ourselves to comparisons between two data points or between one data point and a constant. We consider data points representing the number of confirmed cases in specific regions and at specific times, the number of recoveries, the number of deaths, and the ratio of deaths to total cases. As comparison operators, we consider equalities and inequalities. Non-terminal symbol `<F>` translates into a numerical value that appears in the input claim text. Symbol `<Country>` translates to the name (i.e., a string) of a country that appears in the database.

Note that (unlike for elements that appear in the query template representation) there is no one-to-one mapping between transformation rules in each of the two grammars and choice points in the query space (for which questions to crowd workers are generated and classifiers are introduced). Instead, in case of IEA, as implied by the associated template from Example 3, we have four choice points, referring to formula, tables, years, and primary key values. In case of CoronaCheck, our choice points include comparison formula and data source, as well as time and country.

A.5 System Dimensions

Type	Dimension	AggCheck	Tapas	TabFact	Scrutinizer
Input	Implicit Claims		✓	✓	✓
	Schema-Independence	✓	✓	✓̸	
	Multi-variable Formulas	✓̸	✓̸		✓
	Multi-tables	✓̸			✓
Output	Interpretability	✓	✓̸		✓
	Alternative Interpretations	✓			✓

Table A.4: Dimensions that characterize the systems (✓̸ denotes partial support).

We believe that, given the increasing number of fact-checking systems, it is important to start characterizing them with clear dimensions to enable a more rigorous comparison. We first describe four main dimensions that characterize the input across the different proposals. Then, we discuss two dimensions that characterize the output. A summary of the dimensions and how systems support them is reported in Table A.4.

A.5.1 Input Dimensions

Explicit claims are handled by all fact-checking systems, as they are much easier to deal with. However, the support for **Implicit Claims**, i.e. statistical claims that do not mention a number and can be verified by a Boolean function that takes as parameters cell values in the input relation, requires a deeper understanding of the semantics behind the given sentence. One approach to dealing with this problem is feature-based entity linking, where all entities are detected in the input statement and a set of pre-defined trigger-words are used to build programs representing the semantics of the statement (Chen et al., 2020c). However, such approaches are very sensitive to the error-prone entity linking process. Another approach is to learn such implicit claims in a supervised manner. Scrutinizer learns from the classifiers' labels (Karagiannis et al., 2020). Tapas also learns correlations between the text and the table during the pre-training process.

Another dimension is **Schema-Independence**. AggCheck, TabFact and Tapas can consume potentially any table with any unseen schema, while Scrutinizer is limited to tables whose row index values and attribute labels have been trained on. For Scrutinizer, adding new tables requires fine-tuning the classifiers. The operation is not expensive in terms of execution time, because its classifiers are based on a fine-tuning procedure, rather than having to pre-train again from scratch; however, it requires specific annotations that go beyond the true/false binary label. This dimension highlights that Scrutinizer is domain-specific and thus has to learn the related tables for the task at hand, while AggCheck and Tapas try to be agnostic of the table schema, and can handle any table as input. For TabFact, while it can be used on any unseen schema, our experiments show that it should be trained on the examples at hand in order to obtain

good accuracy performance.

In practice, computations involving values of a database go beyond simple look up and aggregation functions. The function for the verification of a claim can require complex **Multi-variable Formulas**. For example, the Compound Annual Growth Rate[1] is a formula needed to verify a claim in our experiments.

Scrutinizer handles complex formulas on the condition that they are observed in its classifier-specific training data, and resorts to a brute-force approach to assign predicted values to variables. AggCheck can be extended to handle complex aggregation functions. Tapas handles aggregate queries with simple functions where the cell values have been selected by the model. It is not clear if and how Tapas could support functions with more than one variable, and it would require training again the model from scratch such that new functions are learned. Finally, TabFact has no explicit notion of formulas, as it is a black-box model fine-tuned end-to-end on a binary classification task. According to the original paper and our experiments, TabFact struggles to learn how to handle formulas with multiple variables.

TabFact and Tapas assume that the right table to verify the input claim is also given as input. In practice, many tables can be available and the most likely one for the task at hand is identified by Scrutinizer and AggCheck (**Multi-Tables**). Moreover, in some cases more than one table is needed to verify a claim and only Scrutinizer supports verification that requires the combination of values from multiple tables. This dimension highlights one of the limits of the methods that rely on the linearized data fed to the transformers, as it is hard to feed multiple tables without hitting the limit on the size of the input.

A.5.2 Output Dimensions

Interpretability is a key dimension supported by methods that output the query used to verify the claim. However, systems using a black-box model to verify claims, such as TabFact, lack interpretability as an explanation of the prediction is not provided. There do exist methods attempting to explain black-box models which include explanations by simplification (Ribeiro et al., 2016). However, there is no consistent method to define how faithful are the explanations to the model prediction (Jacovi and Goldberg, 2020). Tapas is not fully interpretable since it provides only cell values and, in some cases, the aggregation operation. AggCheck and Scrutinizer expose the declarative query used to verify the associated claim. Systems that predict query fragments and combine them, rather than producing an answer in one shot, are easier to interpret (d'Avila Garcez and Lamb, 2020).

Claims expressed in natural language can be incomplete or ambiguous in many ways. Some systems support **Alternative Interpretations** to clarify how the output changes depending on the details of the verification. Consider a simple claim "Mike scored 30

[1]It describes the net gain or loss of an investment over a certain period of time (https://en.wikipedia.org/wiki/Compound_annual_growth_rate).

points", and a table with two players whose first name is "Mike". The claim is true for one player, but false for the other. AggCheck resolves such ambiguities by evaluating multiple queries and soliciting feedback from users. Scrutinizer learns ambiguities conditioned that they are represented in the training data. Tapas and TabFact do not include any clear means to resolve this kind of ambiguities, as they default to one interpretation in the current architectures. (Veltri et al., 2022) tackle the problem by generating ambiguous training data for models to learn from.

A.6 Proofs

Theorem 1. *Compared to the baseline, relative verification overhead of* Scrutinizer *is at most* $(nop \cdot v_f + nsc \cdot (v_p + s_p))/s_f$.

Proof. Reading through answer options on the final screen adds cost overheads of $nop \cdot v_f$ in the worst case. We have overheads of $nsc \cdot (v_p + s_p)$ for all previous screens. Verifying the claim without help means suggesting a query for the current claim. This has cost s_f in our model. □

Corollary 1. *Setting* $nop = s_f/v_f$ *and* $nsc = s_f/(v_p + s_p)$ *limits verification overheads to factor three.*

Proof. This follows immediately by substituting the proposed formulas in the equations from Theorem 1. □

Theorem 2. *The expected verification cost for answer options* $\langle a_1, \ldots, a_m \rangle$ *is* $v_p \cdot \sum_{i=1..m}(1 - \sum_{1 \leq j < i} p_{a_i})$.

Proof. We consider the case that at most one answer option is accurate (this case is typical). The cost of verifying one answer option is v_p (assuming properties). The probability that workers need to read beyond the i-th option is the probability that none of the first i options is correct: $\Pr(a_1 \text{ to } a_i \text{ incorrect}) = 1 - \sum_{1 \leq j \leq i} p_{a_i}$. The expected cost is the cost of each verification, weighted by the probability that it is necessary: $v_p \cdot \sum_{i=1..m}(1 - \sum_{1 \leq j < i} p_{a_i})$. □

Corollary 2. *Selecting answer options in decreasing order of probability minimizes expected verification cost.*

Proof. Each term in the cost formula, proven in Theorem 2, decreases if the sum of probabilities of the first options increases. Hence, starting with higher probability choices decreases cost. □

Theorem 3. *The pruning power* $\mathcal{P}(S, Q, M)$ *is given by* $\sum_{q \in Q}(1 - \prod_{s \in S} \sum_{i: q \notin E_s^i} \Pr(a_s^i \text{ correct} | M))$.

Proof. The pruning power is given as the expected number of pruned queries: $\sum_{q \in Q} \Pr(q \text{ is pruned})$. Clearly, it is $\Pr(q \text{ pruned}) = 1 - \Pr(q \text{ not pruned})$. Assuming independence, we obtain

$$\Pr(q \text{ not pruned}) = \prod_{s \in S} \Pr(q \text{ not pruned by } s | M)$$

. Assuming mutually exclusive answer options for the same property, we obtain

$$\Pr(q \text{ not pruned by } s) = \sum_{i:q \notin E_s^i} \Pr(a_s^i \text{ correct}|M)$$

. Substitution yields the postulated formula. □

Theorem 4. *Pruning power is sub-modular.*

Proof. Consider the probability that one specific query is not pruned via questions relating to any property, given as $\prod_{s \in S} \Pr(q \text{ not pruned by } s|M)$ (see proof of Theorem 3). From the perspective of each query, adding one more property corresponds to multiplying its probability of not being pruned by a factor between zero and one. For $x_1, x_2, y \in [0, 1]$, it is generally $x_1 - x_1 \cdot y \geq x_2 - x_2 \cdot y$ if $x_1 \geq x_2$. As the probability of not being pruned does not increase when adding questions, the impact of adding a new question on pruning probability decreases for each query. This means the probability of one query of being pruned is sub-modular in the question set. The same applies to pruning power itself (as a sum over sub-modular functions with positive weights is sub-modular). □

Theorem 5. *Using the greedy algorithm, we select a set of questions that achieve pruning power within factor* $1 - 1/e$ *of the optimum.*

Proof. The greedy algorithm is equivalent to the greedy algorithm by Nemhauser (Nemhauser and Wolsey, 1978). The pruning power function is sub-modular (see Theorem 4), it is non-negative (as we sum over probabilities) and non-decreasing (as pruning probability can only increase when adding more questions). Hence, it satisfies the conditions under which those bounds have been proven for Nemhauser's algorithm (Nemhauser and Wolsey, 1978). □

Theorem 6. *Finding optimal question sequences for verifying single claims is in* $O(nsc \cdot npr \cdot nqu)$.

Proof. The greedy algorithm performs $O(nsc)$ steps and considers $O(npr)$ options in each step. Evaluating the pruning power function requires $O(nqu)$ steps. □

Theorem 7. *Claim selection is NP-hard.*

Proof. We prove NP-hardness by a reduction from the knapsack problem. Let $I = \{\langle w_i, b_i\}$ a set of items with associated weights w_i and benefit b_i. The goal is to maximize accumulated benefit $\sum_{i \in I^*} b_i$ for an item set $I^* \subseteq I$ whose accumulated weight remains below a threshold T: $\sum_{i \in I^*} b_i \leq T$. We construct an equivalent instance of claim selection as follows. We introduce an unverified claim c_i for each item $i \in I$. We assume that each claim is located in a separate section (s_i for claim c_i). We set combined verification and reading cost for each claim and associated section to be proportional to item weight: $v(c_i) + r(s_i) = w_i$. Training utility is proportional to benefit ($u(c_i) = b_i$). We choose cardinality bounds that do not influence the solution ($b_l = 0$ and $b_u = |I|$). Now, an optimal solution to claim verification yields an optimal solution to the original knapsack instance (via a polynomial time transformation). □

Theorem 8. *The size of the ILP problem is in $O(cc \cdot sc)$ where cc is the claim count and sc the section count.*

Proof. The number of variables is in $O(cc + sc)$ while the number of constraints (specifically: constraints connecting claims to sections read) is in $O(cc \cdot sc)$. □

B Additional Material for Chapter 4

B.1 More on Reasoning with Soft Rules

Let σ be a signature as in first-order logic. An LP$^{\text{MLN}}$ program Π is a finite set of weighted rules of the form:

$$w : A \leftarrow B \tag{B.1}$$

where A is a disjunction of atoms of σ, B is a conjunction of literals (atoms and negated atoms) of σ, and w is a real number or the symbol α.

When A is $\bot$ (the empty disjunction), the rule asserts that B should be false in the stable model. An LP$^{\text{MLN}}$ rule (B.1) is called *soft* if w is a real number or *hard* if w is α. An LP$^{\text{MLN}}$ program is *ground* if its rules contain no variables. An LP$^{\text{MLN}}$ program Π that contains variables is identified with a ground LP$^{\text{MLN}}$ program $gr_\sigma[\Pi]$, which is obtained from Π by replacing every variable with every ground term of σ. The weight of a ground rule in $gr_\sigma[\Pi]$ is the same as the weight of the corresponding rule in Π. By $\overline{\Pi}$ we denote the unweighted logic program obtained from Π, i.e., $\overline{\Pi} = \{R \mid w : R \in \Pi\}$.

For a ground LP$^{\text{MLN}}$ program Π, Π_I denotes the set of rules $w : R$ in Π such that I satisfies R (denoted $I \models R$) and SM[Π] denotes the set $\{I \mid I$ is a (deterministic) stable model of $\overline{\Pi_I}\}$. The (unnormalized) weight of I under Π is defined as follows:

$$W_\Pi(I) = \begin{cases} exp(\sum\limits_{w:R\in\Pi_I} w) & \text{if } I \in \text{SM}[\Pi]; \\ 0 & \text{otherwise.} \end{cases}$$

The probability of I under Π is the normalized weight defined as follows:

$$P_\Pi(I) = \lim_{\alpha\to\infty} \frac{W_\Pi(I)}{\sum_{J\in\text{SM}[\Pi]} W_\Pi(J)}.$$

In Answer Set programming (ASP), search problems are reduced to computing *stable models* (a.k.a. answer sets), a set of beliefs described by the program. In the case of a

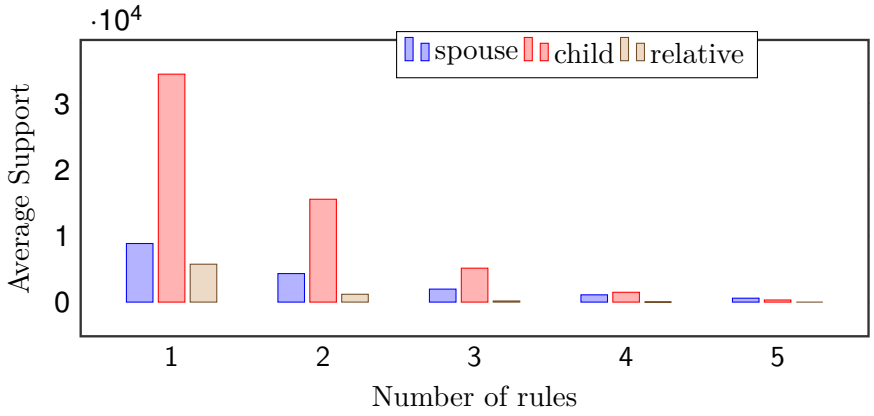

Figure B.1: Support of the overlapping rules.

Horn program, the stable models coincide with the minimal models. LP$^{\text{MLN}}$ programs are transformed to meet the needs of an ASP solver (an, 2014; Lee et al., 2017).

B.2 Rule Support

We designed an experiment to show the impact of increasing the number of overlapping rules on the same target predicate. The goal is to measure how often multiple rules are triggered for the same target triple.

We measure this with the support of a rule, i.e., the number of triples in the knowledge base that satisfy all the atoms in the rule.

To compute the support for more than one rule, we combine the premises of the rules. In this experiment, we picked three predicates (*spouse*, *child* and *relative*), and for each one we selected ten rules randomly. Next, we used DBpedia online endpoint[1] to compute the support for each combination of n (n=1,2,...,5) rules for each predicate. The results in Figure B.1 show that by increasing the number of rules, the support decreases for all predicates. For combinations with more than three rules, the support is very small.

B.3 More Experimental Details

For fine-tuning our models, we use Google Colaboratory (Bisong, 2019), which assigns random GPU clusters of various types. The number of parameters of our models is about 355M. We select the values of our hyper-parameters (shown in Table B.1) on the development sets, by maximizing accuracy.

The execution times vary largely depending on the GPU at hand and on the scenario, with fine-tuning on a Tesla V100 taking from one hour for a single rule to a few hours for all the chaining experiments. The training/validation/testing splits are shown in Table 4.1. Table B.2 shows the sizes of the used test datasets.

[1] http://dbpedia.org/sparql

Hyper-Parameter	Value
Learning Rate	1e-6
Weight Decay	0.1
Number of Epochs	3
Batch Size	16
Learning Rate Decay	Linear
Warm-up Ratio	0.06

Table B.1: Hyper-parameters for fine-tuning our model.

Dataset	Size
Mod0 Test(own)	2,667
Mod1 Test(own)	4,000
Mod2 Test(own)	5,334
Mod3 Test(own)	6,667
Mod4 Test(own)	8,000
Mod5 Test(own)	9,334
Test(D≤5)	9,334
Depth=0	16,057
Depth=1	6,608
Depth=2	5,389
Depth=3	3,993
Depth=4	2,619
Depth=5	1,336

Table B.2: Number of examples in each of the test datasets for the chaining experiment.

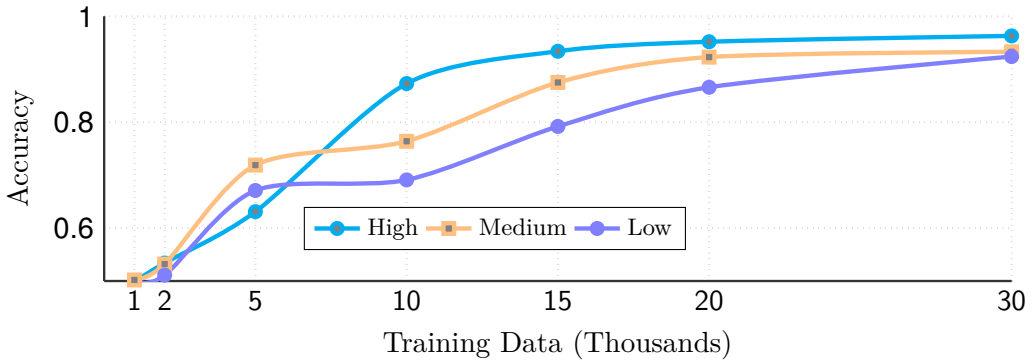

Figure B.2: Impact of the training data size.

B.4 Ablation

B.4.1 Impact of the Data Size

Setting. We report the impact of the size of the fine-tuning data on the model performance. As shown in Table 4.2, the accuracy of the fine-tuned model is higher for rules with higher confidence. We therefore divide the rules in three categories: *High* contains rules with confidence greater than 0.8, *Medium* has rules with confidence between 0.4 and 0.8, and *Low* is for the rest. There are six rules in the *Medium* category and the other two categories have five rules each. For each rule, we fine-tune seven models with 1k, 2k, 5k, 10k, 15k, 20k, and 30k examples.

Results. Figure B.2 shows that having more training data improves the accuracy in all scenarios. For all categories, there is a sizable increase going from 10k to 15k examples; the impact is smaller for higher values. The highest increase is for rules with high confidence, and rules with medium confidence demonstrate larger increase than low confidence.

B.4.2 Role of the Example Format

Setting. When we 'teach' rules to PLMs, we rely on examples with real names from a fixed pool. However, our goal is to 'teach' PLMs the semantics of the soft rule, not the facts in our examples. Thus, we further design an experiment to assess the impact of the format used in the example facts on the behavior of the model. We distinguish two formats for the generated facts: (*i*) real names such as *Alice* and *IBM*, and (*ii*) letters such as *A* and *B*. We first use each format in fine-tuning and we then test both formats. We end up with two test/train scenarios: one with the same format and one with different formats. For this study, we use just one rule: $child(a,c) \wedge parent(c,b) \rightarrow spouse(a,b)$, with 30K examples for fine-tuning, and 2k for testing.

	Train Letter	**Train Name**
Test Letter	.981	.932
Test Name	.977	.985

Table B.3: Impact of the example format on accuracy.

Results. The results in Table B.3 show that the model performance does not depend heavily on using the same fact format for training and testing. With examples using letters in training, the results are slightly better in the case with two formats. We ultimately use names for testing and training in our default configuration, as it yields better results.

B.5 Impact of the Random Seed

Pre-trained transformers often suffer from instability of the results across multiple reruns with different random seeds. This usually happens with small training datasets (an, 2020; Mosbach et al., 2021). In such cases, typically multiple reruns are performed, and the average value over these reruns is reported.

However, the numbers for the main experiments we report are not averaged over multiple reruns as our datasets are considerably large and the models did not suffer from instability due to random seeds. For example, when we reran RuleBERT on a single-rule experiment three times, we obtained accuracy of 0.98959, 0.99551, 0.99636 with a standard deviation of only 0.003.

Yet, for the small dataset *bAbI*, we observed a much higher standard deviation of 0.17. Thus, in this case we report results that are averaged over ten reruns.

B.6 Rule Overlap Example

After generating the data for every rule in Figure 4.2, we generate additional examples using combinations of rules. Below, we show how to handle the interaction of two rules: r_2 and r_3. We follow the procedure in Algorithm 4 by generating facts that trigger the rules, but we only take into consideration hypotheses that deal with rule conclusions.

For example, consider the following facts:

Generated Facts

- f_1: `negparent(Eve,Carl)`
- f_2: `child(Eve,David)`
- f_3: `relative(Eve,David)`
- f_4: `predecessor(Eve,David)`

We can generate an example that triggers two rules: f_2 triggers r_2, and f_3 triggers r_3. Feeding the above facts and rules r_2 and r_3 to the reasoner, we obtain the following output (the numbers in parentheses indicate the likelihood of the triple):

LP$^{\text{MLN}}$ **Reasoner Output O:**

- o_1: `relative(Eve,David) (1.0)`
- o_2: `child(Eve,David) (1.0)`
- o_3: `negparent(Eve,Carl) (1.0)`
- o_4: `spouse(Eve,David) (0.134)`
- o_5: `negspouse(Eve,David) (0.55)`
- o_6: `predecessor(Eve,David) (1.0)`

We produce hypotheses that trigger both rules together. For example, here we generate two hypotheses coming from o_4 and o_5. The confidence (weight) of a hypothesis is given by the LP$^{\text{MLN}}$ reasoner. Taking o_5 as a hypothesis, we feed the following example to the model:

Example #2 (Model Input):

- *Context* : The parent of Eve is not Carl. The child of Eve is David. If the child of the first person is the second person, then the first person is not the spouse of the second person. The relative of Eve is David. If the relative of the first person is the second person, then the first person is the spouse of the second person. The predecessor of Eve is David.
- *Hypothesis* : The spouse of Eve is not David.
- *Weight* : 0.55

We also generate an example, where three rules are triggered: In addition to r_2 and r_3, r_5 is triggered by f_4. We then repeat the same procedure to generate the following example:

Example #3 (Model Input):

- *Context* : The parent of Eve is not Carl. The child of Eve is David. If the child of the first person is the second person, then the first person is not the spouse of the second person. The relative of Eve is David. If the relative of the first person is the second person, then the first person is the spouse of the second person. The predecessor of Eve is David. If the predecessor of the first person is the second person, then the first person is not the spouse of the second person.
- *Hypothesis* : The spouse of Eve is not David.
- $Weight$: 0.6

This procedure is repeated for all combinations of two or more rules. In case when all rules have the same head polarity, we generate a false example by altering the hypothesis and finding the complement of the initial (1-$Weight$) weight. For example, r_2 and r_5 can occur together and both have the same rule head, and thus no conflict occurs. The generated valid example would be as follows:

Example #4 (Model Input):

- *Context* : The parent of Eve is not Carl. The child of Eve is David. If the child of the first person is the second person, then the first person is not the spouse of the second person. The relative of Eve is David. The predecessor of Eve is David. If the predecessor of the first person is the second person, then the first person is not the spouse of the second person.
- *Hypothesis* : The spouse of Eve is not David.
- $Weight$: 0.64

An invalid example is generated from the valid example by altering the hypothesis. Here is an invalid example:

Example #4 (Model Input):

- *Context* : The parent of Eve is not Carl. The child of Eve is David. If the child of the first person is the second person, then the first person is not the spouse of the second person. The relative of Eve is David. The predecessor of Eve is David. If the predecessor of the first person is the second person, then the first person is not the spouse of the second person.
- *Hypothesis* : The spouse of Eve is ~~not~~ David.
- $Weight$: 0.36 (1-0.64)

B.7 Rule Chaining Example

Here is an example that illustrates rule chaining:

Example #5 (Symbolic):

- *Rules* $R =$
  - r_1: `child(A,C)` $\wedge$ `parent(C,B)` $\rightarrow$ `spouse(A,B)`
  - r_2: `child(B,A)` $\rightarrow$ `parent(A,B)`
- *Facts* F :
  - f_1: `negparent(Eve,Carl)`
  - f_2: `child(Bob,Carl)`
  - f_3: `child(Alice,Carl)`
- *Hypothesis* h : `spouse(Alice,Bob)`

f_2 triggers r_2 which produces $t =$

`child(Bob,Carl).`

t and f_3 trigger r_1 to validate the hypothesis h. r_1 and r_2 have been chained to validate the hypothesis. Since we used two rules to validate the hypothesis, we say that this is a chain of depth = 2.

C Additional Material for Chapter 5

C.1 LAMA

Dataset statistics are reported in Table C.1.

	P@1	P@10	P@50	P@100
B	0.223	0.509	0.740	0.845
BTo	0.146	0.327	0.550	0.640
PostTE	.248	.577	.819	.889
BTE	0.291	0.606	0.838	0.899
Top10	0.336	0.660	0.856	0.907
Bot10	0.235	0.534	0.764	0.846
Unif	0.250	0.563	0.798	0.884

Table C.2: Mean over all datasets for every sampling method.

	P@1	P@10	P@50	P@100
Bl	0.245	0.523	0.729	0.811
BlTE	0.297	0.582	0.777	0.849
Rob	0.073	0.235	0.400	0.479
RobTE	0.177	0.331	0.481	0.589

Table C.3: Mean over all datasets for Bert Large (Bl) and Roberta base (Rob) with and without TEs..

Type	Total Size	Dataset	Size	Sample
Country (Co)	3796	P495	896	The Sharon Cuneta Show was created in [MASK] .
		P27	948	Albert II of Belgium is [MASK] citizen .
		P1376	196	Cardiff is the capital of [MASK] .
		P1001	665	National Congress of Honduras is a legal term in [MASK] .
		P530	174	Vanuatu maintains diplomatic relations with [MASK] .
		P17	917	Cairo American College is located in [MASK] .
Football Position (FP)	737	P413	737	Curt Flood plays in [MASK] position .
Manufacturer (Ma)	878	P176	878	iPod shuffle is produced by [MASK] .
Organization (Org)	837	P108	342	David Dimbleby works for [MASK] .
		P178	495	iPod Classic is developed by [MASK] .
Occupation (Occ)	915	P106	915	Murray Grand is a [MASK] by profession .
Year GRE (Y (GRE))	1821	date_of_birth	1821	Emily Ballou (born [MASK]).
Genre (Ge)	849	P136	849	Boyd Raeburn plays [MASK] music .
Group (Gr)	212	P463	212	Russian Football Union is a member of [MASK] .
Language (L)	4118	P407	756	The Pirate Bay was written in [MASK] .
		P103	954	The native language of Jan Davidsz. de Heem is [MASK] .
		P1412	921	Leone Caetani used to communicate in [MASK] .
		P37	707	The official language of Iitti is [MASK] .
		P364	780	The original language of Do Phool is [MASK] .
Specialization (Sp)	533	P101	533	John Archibald Wheeler works in the field of [MASK] .
Religious Position (RelP)	727	P39	727	John Joseph Williams has the position of [MASK] .
Record Label (Rec)	256	P264	256	Amr Mostafa is represented by music label [MASK] .
City (Ci (GRE))	3689	place_of_birth	2925	Jacques Autreau was born in [MASK] .
		place_of_death	764	Robert Jack died in [MASK] .
City (Ci)	6490	P131	774	Saharsa district is located in [MASK] .
		P20	844	Fredegund died in [MASK] .
		P937	864	Xavier Zubiri used to work in [MASK] .
		P19	704	James Jackson Putnam was born in [MASK] .
		P740	643	Standard Bank was founded in [MASK] .
		P190	283	Inverness and [MASK] are twin cities .
		P36	400	The capital of Realm of Stefan Dragutin is [MASK] .
		P159	700	The headquarter of Shelbourne F.C. is in [MASK] .
		P47	542	Campi Bisenzio shares border with [MASK] .
		P276	736	Hiroshima International Animation Festival is located in [MASK] .
Continent (Con)	964	P30	964	Dominion Range is located in [MASK] .
Musical Instrument MI	739	P1303	739	Kerry King plays [MASK] .
TV Network (TVN)	806	P449	806	The New Dick Van Dyke Show was originally aired on [MASK] .
Religion (Rel)	452	P140	452	Muhammad Ali Jinnah is affiliated with the [MASK] religion .

Table C.1: LAMA datasets grouped by type. Each dataset belongs to the TREx dataset, unless otherwise stated by (GRE).

Detailed results on the datasets are reported in Table C.4. A full inference run on all LAMA datasets takes on average approximately 5 minutes on Google Colab with a Tesla P100 with a batch size of 32. We vary λ from 0 to 5. We repeat the experiment in Section 5.5.1 with every sampling strategy and report results in Table C.5. For LANGUAGE, all sampling methods outperform the weighted method. This is due to the non-optimal value of λ produced for one dataset that reduced the average value. Indeed, setting a more suitable value for λ, pushes the precision scores comparably to other sampling methods. Surprisingly, for RELIGIOUS POSITION, *Bot10* produces better results on all metrics except $P@1$. This is because most of the golden labels of the data related to religious positions for Christianity, while using *Top10* includes a position for Judaism (rabbi), which is not the case for *Bot10* and *Unif*. Finally, similar results are observed for GROUP and CONTINENT simply because there were less than 10 tokens for each type from the KG.

		P@1	P@10	P@50	P@100
Co	B	0.333	0.578	0.812	0.892
	BTo	0.092	0.269	0.427	0.520
	PostTE	0.323	0.549	0.838	0.888
	BTE	0.393	0.643	0.874	0.916
FP	B	0.003	0.234	0.500	0.701
	BTo	0.203	0.407	0.657	0.730
	PostTE	0.239	0.510	0.883	0.977
	BTE	0.276	0.500	0.826	0.896
Ma	B	0.865	0.945	0.982	0.988
	BTo	0.859	0.939	0.98	0.987
	PostTE	0.008	0.848	0.941	0.965
	BTE	0.564	0.923	0.970	0.978
Org	B	0.347	0.733	0.928	0.961
	BTo	0.248	0.393	0.447	0.464
	PostTE	0.347	0.733	0.928	0.961
	BTE	0.279	0.730	0.955	0.977
Occ	B	0.002	0.089	0.463	0.839
	BTo	0.0	0.023	0.305	0.441
	PostTE	0.012	0.188	0.619	0.951
	BTE	0.036	0.196	0.601	0.904
Y (GRE)	B	0.016	0.152	0.623	0.806
	BTo	0.002	0.102	0.499	0.783
	PostTE	0.016	0.152	0.623	0.806
	BTE	0.017	0.146	0.624	0.802
Ge	B	0.007	0.470	0.697	0.803
	BTo	0.0	0.087	0.636	0.743
	PostTE	0.594	0.690	0.834	0.844
	BTE	0.589	0.686	0.831	0.841
Gr	B	0.692	0.821	0.861	0.886
	BTo	0.025	0.214	0.652	0.801
	PostTE	0.692	0.821	0.861	0.886
	BTE	0.692	0.821	0.861	0.886
L	B	0.600	0.892	0.971	0.988
	BTo	0.168	0.436	0.622	0.716
	PostTE	0.445	0.750	0.965	0.982
	BTE	0.556	0.855	0.967	0.980
Sp	B	0.085	0.362	0.569	0.688
	BTo	0.0	0.008	0.138	0.291
	PostTE	0.085	0.362	0.569	0.688
	BTE	0.097	0.302	0.545	0.640
RelP	B	0.070	0.281	0.709	0.900
	BTo	0.0	0.023	0.219	0.372
	PostTE	0.010	0.503	0.938	0.962
	BTE	0.159	0.488	0.951	0.959
Rec	B	0.140	0.416	0.733	0.877
	BTo	0.062	0.342	0.539	0.642
	PostTE	0.095	0.218	0.539	0.621
	BTE	0.152	0.444	0.819	0.922
Ci (GRE)	B	0.142	0.353	0.566	0.657
	BTo	0.0	0.003	0.024	0.06
	PostTE	0.142	0.353	0.566	0.657
	BTE	0.144	0.365	0.572	0.663
Ci	B	0.287	0.570	0.757	0.831
	BTo	0.058	0.107	0.194	0.252
	PostTE	0.284	0.563	0.756	0.839
	BTE	0.299	0.577	0.768	0.849
Con	B	0.216	0.481	0.730	0.822
	BTo	0.613	0.864	0.969	0.995
	PostTE	0.477	0.885	0.963	0.995
	BTE	0.408	0.882	0.945	0.980
MI	B	0.064	0.390	0.553	0.614
	BTo	0.131	0.64	0.821	0.832
	PostTE	0.017	0.587	1.000	1.000
	BTE	0.017	0.593	0.991	1.000
TVN	B	0.210	0.858	0.986	0.997
	BTo	0.161	0.609	0.952	0.992
	PostTE	0.210	0.858	0.986	0.997
	BTE	0.200	0.877	0.993	0.997
Rel	B	0.107	0.536	0.883	0.967
	BTo	0.002	0.422	0.814	0.900
	PostTE	0.476	0.809	0.925	0.981
	BTE	0.352	0.876	0.998	1.000

Table C.4: Average precision scores for different types of the LAMA dataset for BERT (B), BERT with additional explicit type token (BTo), TE applied at the output (PostTE), and BERT with TE (BTE).

		P@1	P@10	P@50	P@100
Co	Top10	0.407	0.708	0.891	0.930
	Bot10	0.326	0.608	0.836	0.903
	Unif	0.355	0.601	0.83	0.904
FP	Top10	0.277	0.564	0.861	0.91
	Bot10	0.0	0.04	0.684	0.746
	Unif	0.223	0.561	0.859	0.911
Ma	Top10	0.770	0.921	0.974	0.984
	Bot10	0.865	0.945	0.982	0.988
	Unif	0.865	0.945	0.982	0.988
Org	Top10	0.606	0.866	0.966	0.979
	Bot10	0.307	0.665	0.906	0.951
	Unif	0.275	0.588	0.897	0.951
Occ	Top10	0.087	0.547	0.849	0.921
	Bot10	0.002	0.089	0.48	0.845
	Unif	0.001	0.089	0.496	0.872
Y (GRE)	Top10	0.019	0.145	0.618	0.792
	Bot10	0.010	0.109	0.377	0.578
	Unif	0.018	0.147	0.625	0.803
Ge	Top10	0.582	0.703	0.84	0.842
	Bot10	0.006	0.416	0.703	0.806
	Unif	0.043	0.057	0.356	0.636
Gr	Top10	0.692	0.821	0.861	0.886
	Bot10	0.692	0.821	0.861	0.886
	Unif	0.692	0.821	0.861	0.886
L	Top10	0.61	0.905	0.981	0.992
	Bot10	0.612	0.894	0.975	0.989
	Unif	0.603	0.895	0.976	0.993
Sp	Top10	0.085	0.354	0.573	0.682
	Bot10	0.081	0.356	0.577	0.688
	Unif	0.087	0.391	0.581	0.686
RelP	Top10	0.300	0.600	0.955	0.959
	Bot10	0.0	0.651	0.961	0.965
	Unif	0.0	0.506	0.954	0.962
Rec	Top10	0.193	0.469	0.885	0.942
	Bot10	0.144	0.428	0.761	0.881
	Unif	0.148	0.436	0.761	0.881
Ci (GRE)	Top10	0.148	0.37	0.584	0.678
	Bot10	0.098	0.320	0.521	0.613
	Unif	0.084	0.309	0.525	0.628
Ci	Top10	0.306	0.580	0.773	0.851
	Bot10	0.280	0.548	0.734	0.805
	Unif	0.292	0.578	0.762	0.841
Con	Top10	0.408	0.882	0.945	0.980
	Bot10	0.408	0.882	0.945	0.980
	Unif	0.408	0.882	0.945	0.980
MI	Top10	0.074	0.699	0.849	0.996
	Bot10	0.064	0.39	0.553	0.614
	Unif	0.017	0.692	0.996	1.000
TVN	Top10	0.086	0.864	0.997	0.997
	Bot10	0.225	0.864	0.984	0.997
	Unif	0.299	0.897	0.986	0.997
Rel	Top10	0.396	0.879	0.998	1.000
	Bot10	0.119	0.590	0.914	0.988
	Unif	0.089	0.734	0.972	0.988

Table C.5: Average precision scores for different types of the LAMA dataset with various sampling methods to compute the TE.

C.2 Generated Text

A few examples of generated text with TEs are reported in Table C.6.

	Generation
GPT-2	I like a lot of the things that your parents did at the age of 9 that made you happy, and now you're in my life."The former wrestler said he is glad he didn't pass that test. "They thought I was a bad idea, as far as it goes in life," Mayweather said of his parents and co-stars. "But that went way overboard for me. I went from having a bad childhood and now living in the Bronx with my aunt and her little sister to a guy who's trying to put it all in the right place."That brought me back to the top of my wrestling career and I think the most special person is David Letterman, who gave me that opportunity...
GPT2 + COUNTRY TE	I like a lot of the things that your guys are doing in the dressing room that the team does, and that's what we do. Q: You have said in a couple of different emails that you wanted to go back to **Switzerland**, but that's a different subject as well. A: Well, you will return. I have spoken to the players and said, that way, to go back to **France** and see the success. Q: What's the biggest point of contention from last week? A: I was on the bench in the first half of the game against **France**, although today I was on the bench. But yesterday I was in the team too. It was a big point of contention. And then tomorrow I will go to **Switzerland** . I had a talk with the coaches this morning and they said to him there were plenty people in the dressing room. Q: Will the **U.S.** play at the **Emirates** ? A: That's a possibility for sure...
GPT-2 + YEAR TE	I like a lot of the things that she is doing at the moment, but that is for the time being." She also said she has a lot of work ahead in mind when he arrives on an episode of "The Simpsons." For a fuller picture of her as a show host and actor see "What Happened to the Kids on the Island of the Gummy"? A documentary record (**1990**), also by David R. Seidman. Also on H&M Records. The last time she appeared on a TV show was on "Saturday Night Live" in **2007** , and in "My Brother's Wedding" on ABC in August **2007** . The first time she appeared on "The Tonight Show, with Johnny Carson" was in **1996** . She also won the Tony Award with "The Little Princess" from the Writers Guild and the Outstanding Dramas Award in **2002** . When interviewed on the TV show she recalled that she has worked as the 'Gummy' and 'Grandma' of the show, and gave the following list of roles for "My Brother's Wedding, with Johnny Carson" of her own first year. The '**90s** came in the last season of "How I Met Your Mother" which she co-starred on...
GPT-2	I hate *RelG*, yet *RelG* are the ones who are allowed to carry out this murderous act by terrorising *RelG* in *Religious Country* and then going on to kill innocent *RelG* in the future."...
GPT-2 + AFFECTION TE	I hate *RelG*, yet *RelG* are the ones who are allowed to carry the torch of *Religion* and fight injustice and oppression. You and I will continue to fight this oppression till the end and in the long run, in every age we are going forward. You will do whatever is necessary to keep our sisters on the right path...

Table C.6: We report a set of truncated generated texts using GPT-2 and how they change using TEs for COUNTRY and YEAR. We also try to remove hate speech using an AFFECTION embedding derived from keywords such as love and cheerful. We replace specific keywords by more general keywords. *RelG* stands for a religious group that has been used in the underlined prompt and has been hidden for ethical considerations.

D Additional Material for Chapter 6

D.1 Note and Rating Questions

Input	Possible Output
Given current evidence, I believe this tweet is:	(1) NOT MISLEADING (2) MISINFORMED OR POTENTIALLY MISLEADING
If this tweet were widely spread, its message would likely be believed by:	(1) BELIEVABLE BY FEW (2) BELIEVABLE BY MANY
If many believed this tweet, it might cause:	(1) LITTLE HARM (2) CONSIDERABLE HARM
Finding and understanding the correct information would be:	(1) EASY (2) CHALLENGING
Is this tweet misleading because it contains a factual error?	True, False
Is this tweet misleading because it contains a digitally altered photo or video?	True, False
Is this tweet misleading because it contains outdated information that may be misleading ?	True, False
Is this tweet misleading because it is a misrepresentation or missing important context?	True, False
Is this tweet misleading because it presents an unverified claim as a fact?	True, False
Is this tweet misleading because it is a joke or satire that might be misinterpreted as a fact?	True, False
Is this tweet misleading for other reasons ?	True, False
Is this tweet not misleading because it expresses a factually correct claim?	True, False
Is this tweet not misleading because it was correct when written, but is out of date now?	True, False
Is this tweet not misleading because it is clearly satirical/joking ?	True, False
Is this tweet not misleading because it expresses a personal opinion?	True, False
Is this tweet not misleading for other reasons ?	True, False
Did you link to sources you believe most people would consider trustworthy?	Yes, No

Table D.1: Questions to Birdwatch participants when writing a note for a tweet.

Input	**Possible Output**
Do you agree with the note's conclusion?	Yes,No
Is this note helpful?	(1) NOT HELPFUL (2) SOMEWHAT HELPFUL (3) HELPFUL
Is this note helpful because it was clear and/or well-written?	Yes,No
Is this note helpful because it cites high-quality sources?	Yes,No
Is this note helpful because it directly addresses the Tweet's claim?	Yes,No
Is this note helpful because it provides a neutral or unbiased language?	Yes,No
Is this note helpful for other reasons ?	Yes,No
Is this note unhelpful because it contains incorrect information" ?	Yes,No
Is this note unhelpful because there are sources missing or unreliable?	Yes,No
Is this note unhelpful because it misses key points or irrelevant?	Yes,No
Is this note unhelpful because it is hard to understand?	Yes,No
Is this note unhelpful because it contains an argumentative or biased language?	Yes,No
Is this note unhelpful because it contains spam, harassment, or abuse?	Yes,No
Is this note unhelpful because the sources do not support note?	Yes,No
Is this note unhelpful because it is an opinion or speculation?	Yes,No
Is this note unhelpful because it is not needed on this Tweet?	Yes,No
Is this note unhelpful for other reasons ?	Yes,No

Table D.2: Questions to Birdwatch participant when rating a note.