Towards a Benchmark for Fact Checking with Knowledge Bases

Viet-Phi Huynh Paolo Papotti
EURECOM, France

ABSTRACT

Fact checking is the task of determining if a given claim holds. Several algorithms have been developed to check facts with reference information in the form of knowledge bases. While individual algorithms have been experimentally evaluated, we provide a first publicly available benchmark evaluating fact checking implementations across a range of assumptions about the properties of the facts and the reference data. We used our benchmark to compare algorithms designed on different principles and assumptions, as well as algorithms that can solve similar tasks developed in closely related communities. Our evaluation provided us with a number of new insights concerning the factors that impact the performance of the different methods.

ACM Reference Format:
Viet-Phi Huynh Paolo Papotti EURECOM, France. 2018. Towards a Benchmark for Fact Checking with Knowledge Bases. In *Proceedings of Reasoning on Data Workshop at the Web Conference 2018 (Reasoning on Data'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/

1 INTRODUCTION

Fact checking refers to the task of verification of textual content. Given the increase of incorrect claims over the Web, fact checking is no longer an activity for journalists only. To tackle the spread of misleading information, Web companies have introduced forms of fact checking in their services. In these approaches, information on the trust of the news is gathered from websites (e.g., politifact.com, factcheck.org, and snopes.com), where journalists manually assess the quality of the information.

However, the proliferation of websites and bots spreading false information has motivated an effort for computational fact checking [4], which aims at automatic verification to scale over thousands of daily facts. Several algorithms focus on different types of facts and different domains. We are interested here in techniques that focus on validating "worth checking" facts against trustful Knowledge Bases (KBs) [3, 9]. In the following, we assume the facts and the entities involved have been identified [8], and focus our attention on the step estimating the *veracity* of a given fact (expressed as structured data) w.r.t. reference data considered trusted.

The core problem for fact checking with KBs is that we cannot assume the reference information to be complete (Open World Assumption), i.e., we cannot say if a fact not in the KB is false or just missing [5]. Given a KB K and a fact f, the fact checking algorithms should therefore state if f is a valid missing fact in K.

There are many factors that affect the outcome of the fact checking step. The quality of the reference KB plays a pivotal role, and it must be selected according to the specific domain. Once the KB has been fixed, there is the challenge of picking the right algorithm to validate the facts. There are several proposals available, that differ significantly in their principles and assumptions. Also, given the novelty of the field and the complexity of the problem, there are still no established and unified sets of metrics and datasets.

In this work, we lay the foundation for a benchmark to compare and contrast fact checking algorithms that rely on external information in the form of RDF KBs. Our ultimate goal is to create a benchmark with a large variety of annotated datasets, tools to create synthetic datasets with different properties, and metrics to evaluate different algorithms on a level playing field.

Our contributions in this work are the following.

1. We classify most of the existing fact checking algorithms by their methodology and discuss their main properties (Section 2).

2. We craft the datasets and the metrics for a fair evaluation of the methods in an early version of our benchmark (Section 3).

3. We demonstrate the use of the benchmark[1] with an experimental analysis of representative algorithms from our classification. (Section 4).

2 ALGORITHM CLASSIFICATION

We first give some background and fix the terminology. We then classify different fact checking algorithms according the methods they use to solve the problem.

2.1 Background

A *fact* is defined as a triple that has the form of ("subject" s, "predicate" p, "object" o). Natural language processing (NLP) techniques are used to convert a claim in natural language into a structured format. Facts can be classified into categories, such as numerical, quote, and object property. We focus on object properties, which are facts stating a relationship between the subject and the object in a sentence, e.g., Sacramento is the capital of California.

A *Knowledge Base* (KB) is a direct graph where nodes correspond to entities (subject or object in a fact) and edges correspond to binary predicates among entities. We focus on algorithms that take as input a KB and a fact that is not part of it. Such algorithms assess if the given fact belongs to the missing part of the KB (therefore is "true") or no (is "false").

Methods assume that training examples (labelled facts) are available to build the models. We will detail how we craft our datasets in terms of training data, but, in general, the common assumption is that the KB is trustable, and examples are extracted from it.

2.2 Path Based Algorithms

Given a fact (s, p, o), this group of algorithms makes a decision for it by exploiting the paths in the KB of existing p triples. These KB triples act as positive examples for the learning of the alternative paths (different from p) between their subjects and objects. Properties of the paths are then modeled as features in a classifier that decides if predicate p holds for the given s and o.

2.2.1 Knowledge Linker (KL). This algorithm builds an internal model based on a weighted adjacency matrix with edge weights

[1] Code and data available at https://github.com/huynhvp/Benchmark_Fact_Checking

computed as the in-degree of each node in the KB, then transformed to similarity scores [3]. As the model ignores the labels (semantics) of the predicates, it evaluates the validity of an input fact based on measuring the proximity between its subject and object. Two distance closures are introduced. With the *Metric closure*, every path connecting a given subject and object is mapped to a score computed on the generality of the nodes in the path, where the generality of a node is its frequency in the KB. The more often the node occurs in KB, the less information it conveys. With the *Ultra-metric closure*, only the node with highest generality is used to compute the score for each possible path. In both cases, the maximum score is equivalent to the shortest path between subject and object.

2.2.2 Discriminate Predicate Path Mining (KG-Miner). This algorithm identifies and exploits frequent anchored predicate paths between pair of entities in the KB [9]. Given an input fact (*s, p, o*), it collects as training data the predicate paths for node pairs that satisfy p in the KB and have subject with the same type of s (denoted $\varphi(s)$) and object with type of o (denoted $\varphi(o)$). From each subject $u \in \varphi(s)$ and corresponding object $v \in \varphi(o)$, predicate paths that alternatively represent predicate p are extracted from the KB with a depth-first search (DFS) traversing the graph from u to v up to a length of m. The information gain of these paths and corresponding labels is computed based on their number of occurrences. It then selects the most discriminative paths and plugs them into training for a logistic regression model that optimizes the Area Under Receiver Operating Characteristic (AUROC). This model is then used to compute the likelihood of an input fact.

2.2.3 Path Ranking Algorithm (PRA). PRA discovers the relationships among entities in a stochastic way by performing random walk inference over the KB [7]. For the feature extraction, from a given training set of triples, it uses *two-sided, unconstrained* random walk starting at the source and corresponding target nodes to retrieve paths connected between them. Top k paths for each training instance are kept based on their number of occurrences and are collected into a feature matrix. A value in the matrix corresponding to a training instance (*s, p, o*) is the probability of arriving at the target node o by a random walk starting at source node s and following a specific path among its top k paths. This probability is computed using the approximate method of rejection sampling to reduce the computational complexity. The feature matrix is then used with a classifier to validate the input fact.

2.3 Sub-Graph Based Algorithms

Given a fact, this method extends the previous approach by using sub-graphs of the KB that model the subjects and the objects of the examples. These sub-graphs can be built in different ways, and their features are used in a classifier to evaluate input facts.

2.3.1 Sub-graph Feature Extraction (SFE). SFE [6] extracts features from sub-graphs built from the nodes in the KB. Such features are then used in a classifier to test input facts. While KG-Miner uses a full-graph exhaustive search to extract features, which is costly with complexity proportional to the out-degree to the power of path length per node, PRA uses random walk, which reduces the computational time, but leads to an approximation. To avoid these issues, SFE provides a more efficient way to extract features by using a sub-graph breadth-first search (BFS).

Given a parameter m, the *sub-graph of depth m* for each node n is the result of m BFS steps starting at it. Features are then extracted with two variants. In the first one (Predicate Path), it uses a sequence of predicates that connect a source node to target node by intersecting the sub-graphs of source and target on intermediate sharing nodes (similar to KG-Miner and PRA). The alternative (one-sided) uses a sequence of predicates that starts at source node or target node, but does not necessary reach at corresponding target or source node. SFE uses *binary features*: it disregards the frequency (like KG-Miner) or the probability (like PRA) of feature paths.

2.4 Embedding Based Algorithms

Embeddings encode entities and relations in the KB into a low-dimensional vector space while preserving certain information of the graph and minimizing a margin-based ranking loss. The idea is that a relation in the graph can be interpreted as a translation from subject entity to object entity in the embedding space.

To check a fact (*s, p, o*), this method checks the relevance of the embedding representations $\mathbf{s}$ of s and $\mathbf{o}$ of o with respect to embedding representation $\mathbf{p}$ of relation p though a specific score function *f(s, p, o)*. The score function f depends on the projection used to transform entities and relations.

2.4.1 Knowledge Graph Embedding (Para_GraphE). TransE [2] represents a relation p from triple (*s, p, o*) as a translation from subject s to object o on the same low-dimensional embedding space, that is $\mathbf{s} + \mathbf{p} \approx \mathbf{o}$ if (*s, p, o*) is true. Its score function is defined as: $f(s, p, o) = \|\mathbf{s} + \mathbf{p} - \mathbf{o}\|$, where $\|.\|$ can be either L1 or L2 norm.

TransE has drawbacks when dealing with non functional relations. As it uses the same embedding space for both entities and relations, a many-to-one relation, for example, requires different subject entities in this relation to have identical embedding representation, which is not a good assumption. To address this issue, TransH enables an entity to have different embedding representations w.r.t. the different relations it participates[10]. For each relation p, it introduces a relation-specific hyperplane w_p (normal vector) and defines embedding vector $\mathbf{p}$ on this hyperplane.

Embedding learning in a large-scale knowledge graph is an expensive operation. We use a framework that offers multi-thread implementation of embedding algorithms [11].

There are other methods that can be adapted to verify facts with KBs, such as logical rules mined from the KB itself [5]. As discovered rules needs to be manually verified, we leave the study of algorithms that require manual setup to future work.

3 THE BENCHMARK

Knowledge Graph. We instantiate our benchmark with DBPedia [1], an RDF knowledge base with triples extracted from Wikipedia. From these triples, we construct a graph by assigning each unique entity to a graph node, and converting the triple into a directed edge with label "predicate" from the entity "subject" to entity "object". We obtain a directed graph with ≈4M nodes, ≈27M edges, and 671 relations. Other KBs can be plugged to the benchmark as long as they are expressed according to the graph format.

	Dataset	Example Facts	True / Total
Functional	Capital 1	Arizona, capital, Phoenix	50 / 300
		Arizona, capital, Oregon	
	Capital 2	Massachusetts, capital, Boston	50 / 259
		Massachusetts, capital, Worcester	
	Capital 3	Massachusetts, capital, Boston	50 / 259
		Massachusetts, capital, Worcester	
		France, capital, Paris*	
		Japan, capital, Tokyo*	
		*Japan, capital, Osaka**	
Non functional	Bestseller	Never Go Back, author, Lee Child	93 / 558
		Personal (novel), author, Lee Child	
		Greater Journey, author, Michael Lewis	
	Award	J. Kittinger, award, War Prisoner Medal	100 / 600
		J. Kittinger, award, Bronze Star Medal	
		H. F. Davison, award, Military Cross	
		J. L. Morgan, award, Military Cross	
		Lothar Linke, award, 2009 RTHK	

* The instances used only for training.

Table 1: Datasets in the benchmark (false claims in italic)

Test cases. We collect and extend test cases used for fact checking in the literature. Each dataset includes both true and false facts for a specific relation, as shown in Table 1. We distinguish functional and nonfunctional relations, and craft datasets that have different properties:

Capital 1. A one-to-one relation, which contains facts *capitalOf(city,state)* for 50 US states. From these 50 correct capital city-state triples, we create 200 incorrect triples by random matching each capital to 4 other states for a total of 200 statements (50 true and 250 false).

Capital 2. A one-to-one relation, which contains again true facts for relation *capitalOf* for 50 US states but with *challenging false facts*. Unlike **Capital 1**, in which false instances are created by random matching capital cities to states, in **Capital 2** contains as false facts triples extracted from relation *largestCities*, which is semantically close to *capitalOf*. For each state in the 50 correct capital-state triples, we include its largest cities in the test set for a total of 209 false facts.

Capital 3. A one-to-one relation, which focuses on checking relation *capitalOf* of 50 US states but with *heterogeneous positive examples in training*. In capital datasets **1** and **2**, training entities are semantically close to entities in testing phase. That is, to check whether a US city is capital of a US state, the methods rely on the information from other US capitals and cities. In this dataset, we train the algorithms with capital of countries in the world. Now, the examples model a more general concept of *capital*, as it is no longer simply a US capital city of a US state. We employ the same 259 true/false instances as **Capital 2** for testing stage, but training stage includes worldwide facts, such as Paris-France as true labels, and Osaka-Japan as false labels.

Bestseller. A one-to-many relation, where we focus on the persons in the *author* relation with at least one book. We collected 63 authors who wrote 93 books that appeared on New York Time bestseller list between 2010 and 2015 (an author can have more than one best-seller book). We create 465 incorrect pairs by random matching each book to 4 other authors.

Award. A many-to-many relation, for which we check statements between persons who won prizes as an *award*. We consider 50 persons who are awarded 69 prizes from different fields (military, sport, art). We create 500 negative-labeled pairs by random matching each person to 5 other prizes.

Evaluation metric. We use the Area Under the Receiver Operating Characteristic curve (AUROC) as a metric to evaluate the algorithms. Our datasets have a low proportion of true statements and AUROC is more informative than classification accuracy when dealing with high class imbalance.

4 ALGORITHM ANALYSIS

The goal of this section is to show that the right benchmark can lead to useful insights about the advantages and limits of the different methods. We therefore executed different algorithms with our datasets to better understand their behavior and the current state of the art.

Experiment setting. We test KL with metric and ultra-metric closure [3]. KG Miner and SFE are tested with the maximum predicate path length m equals to 3 (default value [9]), 4, and 5. PRA is set with number of random walk per source node w/s = 100 and 200, number of random walk per feature path w/p = 50 (default value [7]) and 100. For TransE, TransH, we set 100 as embedding dimension and implement embedding learning with a margin of one and a learning rate of 0.01 for 1,000 epochs, as recommended [2].

All experiments are performed on a on machine with 8 1.8Ghz CPUs and 16GB RAM using 10-fold cross validation, except for **Capital 3**, in which the testing set is different from the training one. Source code and datasets of the benchmark can be found at https://github.com/huynhvp/Benchmark_Fact_Checking.

Results and observations. Qualitative results are reported in Table 2. All methods perform well in the simple, one-to-one relation **Capital 1**. Due to the clear false facts and the semantically similar training examples, there are few paths/sub-graphs between 2 entities in false facts and entities in true facts are highly connected. However, differences in the methods become clear with the other datasets involving nonfunctional relations (**Bestseller, Award**) or complex training data (**Capital-2,3**).

KL is more robust to nonfunctional predicates than other path-based methods. The reason is that it does not rely on the predicate semantics as expressed by the labels. However, KL has issues in the classification of **Capital 2**, where false instances are created from other close but different relations. For example, Cambridge is not *capital* of Massachusetts, but it is one of its *largestCities*. Due to the lack of semantics, KL treats both Cambridge and Boston as capital of Massachusetts because it finds good proximity between the two cities and the state in the graph. This leads to worse result, compared to KG-Miner, PRA and SFE.

KG-Miner, PRA, SFE outperform KL and GraphE but show a significant drop in quality for **Capital 2**, where the true *capitalOf* statements are mixed with false statements that relate to *largestCitites* predicate. The two predicates have similar paths that lead to misclassification in some cases. Moreover, **17/50** US capital-cities are also the largest city in their states, which results in degraded performance when the algorithms treat capital as largest city.

Algorithm	Feature	Settings	Capital 1	Capital 2	Capital 3*	Bestseller	Award
KL	Metric		0.98	0.75	0.75	0.82	**0.87**
	Ultra-metric		0.98	0.75	0.75	0.67	0.47
KG-Miner	Predicate path	maxdepth (m) = 3	**1.0**	0.92	0.88	0.78	0.47
		m = 4	Timeout	Timeout	Timeout	Timeout	Timeout
PRA	Random walk path	walk/source (w/s) = 100, walk/path (w/p) = 50	0.98	0.87	0.91	0.83	0.79
		w/s = 200, w/p = 100	0.99	0.91	**0.92**	0.84	0.79
SFE	PRA-style	m = 3	**1.0**	0.93	0.87	0.80	0.46
		m = 5	**1.0**	**0.97**	**0.92**	**0.89**	0.84
Para_graphE	TransE	embedding = 100, r = 0.01	0.84	0.66	0.66	0.65	0.73
	TransH	embedding = 100, r = 0.01	0.82	0.58	0.58	0.64	0.74

* Cross validation not applied for **Capital 3** since we use different testing and training sets

Table 2: Accuracy Performance (AUROC) using 10-fold cross validation

Increasing the length of predicate paths leads to more discriminative/informative feature paths, thereby improving the performance, as we see with SFE from *length=3* (AUROC = 0.93) to *length=5* (AUROC = 0.97). However, this comes with a trade-off in computational complexity, as we discuss later.

KGMiner, PRA, SFE are also challenged by the more intricate training set in **Capital 3**, with the best methods now achieving AUROC = 0.92. This show the *context-dependency* of discriminative path-based models, where a specific relation may have different definition in different contexts. For example, US state capitals and worldwide capitals are different in many ways (such as scale, the institutions they host) and this is reflected by their features.

Non functional predicate **Award** is difficult to model for all methods due to two reasons. First, awards come from very different fields, so it is hard to find a recurrent pattern for most of the people who got at least one. Second, it is hard to define **award** with short predicate paths, as shown by the poor results in *length 3* KG-Miner and SFE. Extracting longer paths bring a significant improvement with *length 5* SFE outperforming *length 3*, as the algorithm can now find informative paths that nicely support checking **Award** facts. Similar comments apply for **Bestseller**, where the impact of the nonfunctional triples is smaller because of the large number of authors with only one book.

TransE and TransH do not achieve the same performance of the other methods. Also, no clear distinctions are visible when comparing their results, despite TransH is expected to better handle non functional relations. This may due to the knowledge graph we used in this work. DBPedia graph is larger and denser than FB15K (≈75K nodes, ≈315K edges), the largest graph used in previous tests [10].

Algorithm	Setting	Time (seconds)
KL	Metric	10.5
KG-Miner	m = 3	0.12
	m = 4	> 600
PRA	w/s = 100, w/p = 50	0.48
	w/s = 200, w/p = 100	0.60
SFE	m = 3	0.001
	m = 5	0.01
Para_graphE	TransE/TransH	4 hours (to learn embed.)

Table 3: Average feature extraction time (sec) for fact.

We report in Table 3 the average execution times for each algorithm to calculate the inference of a fact. Path-based algorithms are faster than computational models such as KL or stochastic models like TransE, TransH, and PRA. As expected, sub-graph search in SFE is more efficient than full-graph search in KG-Miner when increasing the predicate path length.

5 CONCLUSIONS

This work is the first step towards a general benchmark for fact checking algorithms. We plan to extend it with more datasets, including more predicates and more KBs, and more algorithms. An important missing component is a generator of synthetic scenarios that makes use of the lessons learned with our initial datasets. The generator will create scenarios of arbitrary size and complexity by mixing the properties we highlighted, such as challenging false facts and heterogeneous true facts. Another important direction is to extend the test bed with real facts from news, from example from existing fact check challenges (e.g., https://herox.com/factcheck).

REFERENCES

[1] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web* 7, 3 (2009), 154–165.
[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.
[3] 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 (2015), e0128193.
[4] Emilio Ferrara, Onur Varol, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Commun. ACM* 59, 7 (2016), 96–104.
[5] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 413–422.
[6] Matt Gardner and Tom M Mitchell. 2015. Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction.. In *EMNLP*. 1488–1498.
[7] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. (2014).
[8] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. In *KDD*.
[9] Baoxu Shi and Tim Weninger. 2016. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-Based Systems* 104 (2016), 123–133.
[10] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*. 1112–1119.
[11] Wu-Jun Li Xiao-Fan Niu. 2017. ParaGraphE: A Library for Parallel Knowledge Graph Embedding. In *arXiv. 2017*.