Combining Semantic and Linguistic Representations for Media Recommendation

Ismail Harrando[1*] and Raphael Troncy[1]

[1]Data Science Department, EURECOM, Sophia Antipolis, France.

*Corresponding author(s). E-mail(s): ismail.harrando@eurecom.fr;
Contributing authors: raphael.troncy@eurecom.fr;

Abstract

Content-based recommendation systems offer the possibility of promoting media (e.g. posts, videos, podcasts) to users based solely on a representation of the content (i.e. without using any user-related data such as views or interactions between users and items). In this work, we study the potential of using different textual representations (based on the content of the media) and semantic representations (created from a knowledge graph of media metadata). We also show that by using off-the-shelf automatic annotation tools from the Information Extraction literature, we can improve recommendation performance, without any extra cost of training, data collection or annotation. We first evaluate multiple textual content representations on two tasks of recommendation: *user-specific*, which is performed by suggesting new items to the user given a history of interactions, and *item-based*, which is based solely on content relatedness, and is rarely investigated in the literature of recommender systems. We compare how using automatically extracted content (via ASR) compares to using human-written summaries. We then derive a semantic content representation by combining manually created metadata and automatically extracted annotations and we show that Knowledge Graphs, through their embeddings, constitute a great modality to seamlessly integrate extracted knowledge to legacy metadata and can be used to provide good content recommendations. We finally study how combining both semantic and textual representations can lead to superior performance on both recommendation tasks. Our code is available at https://github.com/D2KLab/ka-recsys to support experiment reproducibility.

Keywords: Media Recommendation, Content Representation, Knowledge Graph, Content-based Recommendation, Automatic Annotation, Textual Embeddings

1 Introduction

User engagement with online content has become a crucial element in most if not all content-providing multimedia platforms. Because retaining a user's interest in the provided content and maximizing their time watching/reading/listening to the content directly relates to the profit of these platforms, recommender systems have became an essential component for content-providing services. The role of these recommender systems is to shape and improve the user experience when it comes to consuming and interacting with said content, i.e. help funneling the overwhelming amount of choices into a condensed, targeted and interesting selection of items that the user is most likely to find enjoyable and interesting.

Traditionally, recommendation systems come in two flavours:

- **Collaborative filtering**: leveraging user statistics and their implicit/explicit feedback (views, likes, watch time) to find items to recommend, with the underlying assumption that people who have similar interests interact with the same items.
- **Content-based** recommendations: relying on the content of the item itself to find similar items without any input from the user. Content-based recommendations are particularly interesting in the case of the *cold start problem*, where there is not yet feedback from users (no interactions on the item to based the recommendations on), and in cases where it is hard to collect such feedback (anonymity, privacy).

In this paper, we are interested in the second kind of recommendations which are based solely on the content of the media to recommend. The "content" in content-based can refer to a variety of potential formats: text, image, video. Typically, a representation of such content is extracted or learned, and the task of recommendation is then cast as a content similarity/retrieval task: given the representation of an item of interest (e.g. the video the user is currently watching), and the representation of all items already existing in the catalog, we want to find the items which have the highest similarity to the item of interest. While many varieties of this approach exist (ones that target other metrics such as *serendipity* [1], *diversity* [2] and *explainability* [3]) which may formulate the problem differently, the task, at its core, can be framed as finding the best content representation that allows uncovering a meaningful measure of similarity.

In this paper, we study two dimensions of content-based recommendations. On one hand, we study the performance of multiple off-the-shelf textual representations on the task of recommendations, with a focus on **relatedness**, i.e.

recommendations that are not based on user history, but on an editorial selection of "related content". To the best of our knowledge, this angle is rarely investigated in recommender systems research.

We also posit that the use of Knowledge Graphs (KGs), can capture the high level semantics that the linguistic representations may fail to capture. We start by using human-annotated metadata for the given content, and we show how automatically generated annotations such as extracted topics and recognized named entities, can improve the quality of these representations. In other words, instead of relying only on the textual data, we use several Information Extraction techniques to extract high level descriptors that can automatically create metadata, which in turn can be used to enrich a KG connecting all content in the media catalog. Given the versatility of Knowledge Graphs, they can be suitable to combine these automatic annotations with already existing metadata seamlessly.

To validate this approach, we focus on studying the TED dataset [4], an open-sourced multimedia dataset that offers the unique possibility of evaluating recommendations based on both the content only ("related videos", as curated by human editors) and the user preferences based on their interactions history. We demonstrate that our approach improves the recommendation performance on both tasks, and that KGs are a reliable framework to integrate external knowledge into the task of recommendation. We finally study the possibility of combining the semantic and linguistic modalities, and show empirically that these two modalities are complementary and by combining them, we improve the performance of the recommender system without any added cost of training or collecting user data.

2 Related Work

Content-based Recommender Systems

There is a large and rich literature on content-based recommender systems, where the focus is set on finding representations that convey best the content of the items to recommend. The task of content-based recommendation can be then seen as a task of Information Retrieval, where one document (an item of interest to the user) can be used as a query to find similar content in the catalog of items to recommend. In this work, we are particularly interested in textual content recommendation, and therefore in the various textual representations for this content. Whereas most traditional IR use TF-iDF variants[5], several other approaches to represent the content of textual documents have been proposed in the literature such as topic modeling [6], word embeddings [7–9], and neural models [10]. Transformer-based models are gaining the most traction, as they offer robust and contextualized document representations that simply outperform most existing approaches in most IR tasks, including content similarity. While BERT [11] is the most notable example of this family of methods, the Sentence-BERT [12] variant, which is particularly fine-tuned to generate meaningful sentence embeddings, performs very well on agnostic

content similarity tasks. Beyond textual representations, there is also a growing trend in incorporating external knowledge sources into the content modeling process to generate richer *item descriptions* [13], either via knowledge bases such as `DBPedia`, or semi-structured sources like `Wikipedia`, building towards the notion of Knowledge-based recommender systems [14].

Graph-based Recommender Systems

Given the recent growing interest in Knowledge Graphs and their applications, there is an expanding literature on the techniques and models that can be leveraged to build "knowledge-aware" recommender systems. In their simplest forms, graph embeddings techniques such as TransE [15], when used in conjunction with a "collaborative" knowledge graph containing both descriptions of items and users and the user-item interactions, can be used to generate semantically rich representations that suit the recommendation task [16]. Beyond the collaborative approach, [17] present an approach to bring external knowledge to the task of content-based Knowledge Graphs, identifying two main approaches to what they called "Semantics-aware Recommender Systems" to tackle traditional problems of content-based recommender systems: i) *Top-down Approaches* which incorporate knowledge from ontological resources such as WordNet [18] and encyclopedic knowledge sources such as Wikipedia[1] to enrich the item representations with external world and linguistic knowledge, and ii) *Bottom-up Approaches* which uses linguistic resources such as what we commonly refer to as distributional word representations, e.g. using pre-trained word embeddings to avoid the issue of exact matching in traditional content-based systems. They also raise the problem of the potential use of a graph structure to discover latent connections among items, which we study in our experiments.

More recently, [19] propose a hybrid model, using a transformer-based encoder to represent the textual content of items with an "entity encoding" based on graph embeddings, and combines both as an input to a deep neural network which learns the user-item similarity scoring. [20] offers an extensive survey of Knowledge Graph-based Recommender System approaches, proposing a high-level taxonomy of methods that either use graph embeddings, connectivity patterns (common paths mining), or combining the two. In this paper, we only focus on embedding-based methods to study the use of automatic annotations on the performance of recommender systems. Additionally, unlike some previous works, our work does not tackle the two tasks jointly as a learning problem [21], but attempts to show how the same approach can at the same time improve the performance on both. This work extends previous works that have was shown how automatically extracted metadata can help improving the performance of content based recommendation [22].

[1] https://en.wikipedia.org/

3 Dataset and Metrics

In this section, we describe in detail the dataset that we use for our later experiments and the evaluation metrics that we use to measure the recommender system performance on the two tasks defined below.

3.1 The TED Dataset

The TED dataset [4] is a multimodal dataset which contains the audiovisual recordings of the TED talks downloaded from the official website[2], which sums up to 1149 talks, alongside metadata fields and user profiles with rating and commenting interactions. The metadata fields are as follows: identifier, title, description, speaker name, TED event at which the talk is given, transcript, publication date, filming date, and number of views. For nearly every video, the dataset contains a list of user interactions (marked by the action of "Adding to favorites"), as well as up to three "related videos", which are picked by the editorial staff to be recommended to the user to watch next. What is unique for this dataset is that it provides two sorts of ground truths for the recommender system use-case, that we can formulate in these two tasks:

- **Task 1 - Personalized (user-specific) recommendations (T1)**: based on a user's list of *favorite* talks, the task is to predict what they would watch next. An evaluation dataset can thus be created using a "leave one out" protocol, i.e. removing one interaction from the user list of favorites, and measuring how successful a method is in predicting the omitted item. Most recommender system-type datasets contain a similar information, i.e. what items a user has actually interacted with in reality, based on their viewing/interaction history. This task is usually handled with collaborative filtering methods (e.g. [23]), but is still interesting for content-based recommendation in the case of the *cold start problem*: when a new talk is added to the platform, how can we recommend it to other users? The most common approach is to use its content to recommend it to users who previously liked a similar content.
- **Task 2 - General (content-based) recommendations (T2)**: to the best of our knowledge, this is the only dataset which offers ground truth for multimedia recommendations based on content only, which are referred to as "related videos", manually annotated by TED editorial staff. These are supposed to reflect subjective topical relatedness between talks in the corpus. Performance on this task reflects the model's ability to recommend content to either users without an interactions history (new users, visitors without accounts) or new videos (that have not yet received any interactions). We note that in the ground truth, some talks are associated with three related talks, some with two, and some with only one. We account for this in the evaluation metrics.

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

Previous works have studied specific aspects of this dataset such as sentiment analysis [24], estimating trust from comments polarity and ratings to improve recommendation [25], or studying hybrid recommender systems [26]. In this work, we focus on this dataset as it offers a unique possibility of evaluating content-based recommendation using both real user feedback and hand-picked recommendations, as the later has not been considered in any of the published works on this dataset to the best of our knowledge.

We also note that, while the dataset is multimodal (TED Talks Videos are also available), our work does not tackle visual information extraction, mostly because TED Talks are not visually diverse: they depict speakers and audience in wide or close shots. This is, however, a promising direction of work that has been tackled in previous works [27].

3.2 Metrics

To evaluate the performance of the methods, we use two commonly used metrics in the recommender systems literature: Hit Rate and Mean Reciprocal Rate. In the following paragraphs, T is the number of talks in the dataset, U is the number of users with at least two interactions in their history, K is the number of (ordered) model recommendations to considerate (we picked $K = 10$ in our results), t is a talk ID (which maps to its embedding), u is a user ID (which maps to its embedding, i.e. the average of the embeddings of all talks in the user's history), $rec_j(x)$ is the j^{th} recommendation by a model (x being a user ID for T1 and a talk ID for T2). $hit(x, j) = 1$ if the talk j is indeed in the ground truth for x, otherwise it is 0. $related(t)$ is the number of related talks in T2 (which can be 1, 2 or 3). $rank(x, j)$ is the rank of talk j in the suggested recommendations for talk/user x by descending similarity score.

Hit Rate (HR@K)

A simple metric to quantify the probability of an item in the ground truth to be among the top-K suggestions produced by the system. For **T1**, this means that the left-out item from the user history must be among the K most similar talks to the user embedding (as defined above). For **T2**, this means that the talk that was manually picked by editors is among the K-most similar talks in the embedding space.

For **T1** we get the formula:

$$HR@K = \frac{1}{U} \sum_{t=1}^{U} \sum_{i=1}^{K} hit(u, rec_i(u))$$

For **T2**, we normalize the counting of hits to account for the variance of number of talks in the ground truth so that the Hit Rate is 1 at best (i.e. when all related talks in the ground truth are included in the system's recommendations):

$$HR@K = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{related(t)}\sum_{i=1}^{K} hit(t, rec_i(u))$$

Mean Reciprocal Rate (MRR@K)

Similarly to $HR@K$, this metric also measures the probability of having ground truth recommendations among the system's predictions, but it also accounts for the rank (order) of the prediction: the closest it is to the top of the predictions, the better.

For **T1**, we get the formula:

$$MRR@K = \frac{1}{U}\sum_{t=1}^{U}\sum_{i=1}^{K}\frac{hit(u, rec_i(u))}{rank(u, rec_i(u))}$$

For **T2**, and again to account for varying number of talks in the ground truth, we slightly alter the previous formula so that it is equal to 1 if all related talks are occupying the top spots in the system predictions:

$$MRR@K = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{\sum_{count=1}^{related(t)} 1/count}\sum_{i=1}^{K}\frac{hit(t, rec_i(t))}{rank(t, rec_i(t))}$$

4 Method and Experimental Setup

In this section, we will present the different embeddings (representations) that we briefly mentioned in Section 3.1. As we mentioned in the introductions, we will explore two types of representations: textual representations that we create from the text content of the media (description or transcript), and semantic representations, generated via embedding the nodes of a knowledge graph that we create, first with only the metadata from the dataset, and then with the addition of automatically extracted annotations. For both modalities, an embedding is generated for each talk, allowing us to compute a similarity score between any two talks in the dataset.

4.1 Evaluation Protocol

To evaluate the quality of each representation, we devise an evaluation procedure for each of the two recommendation tasks defined above, as follows:

- For **T1** (user-specific recommendation), we create a test split using the *leave-one-out protocol* that is commonly used in the literature [28], thus having a "training" set which contains all but one talk that the user interacted with (the user has to have at least two interactions otherwise they are dropped). We create a *user embedding* by averaging the computed embeddings (textual or semantic) of all talks in the training set. The top recommendations are then generated by taking the talks which have the highest similarity score

to the user embedding. We note that there is actually no actual training taking place, but this method allows us to leverage actual "historical" user behavior to evaluate purely content-based recommendation.
- For **T2** (item-based recommendation), we consider all "related videos" as a test set. In other words, for each talk, we compute its similarity to all other talks in the dataset, and we recommend the talks which score the highest.

Further details of how to compute the textual and semantic embeddings are provided in Sections 4.2 and 4.3, respectively.

All mentions of similarity refer to the **cosine similarity**, a widely used similarity measure in the representation learning literature, which can be computed as follows:

$$cosine\_sim(u, v) = \frac{u \cdot v}{\|u\|_2 \|v\|_2}$$

Where u and v are the embeddings or representations of the talks to compute the similarity for.

4.2 Textual Representations

In this section, we will explore how several commonly used textual representations (i.e. embeddings) for the task of media content recommendation, with a special emphasis on how they perform on both the task of user-specific recommendation (T1), and content relatedness recommendation (T2).

4.2.1 Textual Embeddings

For our experiments, we select several textual, or document, representations that are commonly used in Information Retrieval (some already introduced in the related work). Some hyperparameter tuning was done on each of the approaches that require it (size of the embeddings, number of training epochs, etc.), and we report only the best performance from each method.

1. **TF-iDF**: for this representation we use `TfidfVectorizer` from the *Scikit Learn* package[3]. We remove any word that appears less than twice.
2. **NMF**: among different topic modeling techniques, NMF performs well [29] and gives a non-sparse document representation, which guarantees that the similarity score between any two talks in the corpus is not zero. We choose the number of topics $N = 300$ and leave all the other parameters at their default configuration as proposed by the Gensim implementation[4].
3. **GloVe**: we use the 300d embeddings pre-trained on the *Wikipedia 2014 + Gigaword* corpus. To create a document representation, we average the embeddings of all its word components (after removing stop words).
4. **FastText**: we use the 300d `fastText` embeddings pre-trained on *Wikipedia + UMBC + statmt.org news* corpus. The document representation is

[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[4] https://radimrehurek.com/gensim/models/nmf.html

obtained by averaging its individual word embeddings (after removing stop words).

5. **Doc2Vec**: here again, we use the Gensim implementation[5], and we train multiple models varying the size of the embeddings, the number of epochs and the window size. The reported results are obtained with the following configuration: 100d embeddings, a window size of 2, and training for 10 epochs.
6. **SentenceBERT**: we use the `sentence-transformers` package[6] with a SentenceBERT model pre-trained on the Natural Language Inference task (`nli-stsb`)[7], which is shown to perform best on the task of textual similarity [12]. No particular preprocessing is done on the text, as contextualized representations are quite robust and can parse text without the need of filtering/lemmatizing.

We note that for the pre-trained word embeddings (*Glove* and *fastText*), we tried with both the simple averaging-word-embeddings methods and the weighted iDF average (i.e. words that appear more in the corpus weigh less in the linear combination of word embeddings). Based on our experimental results, the straightforward averaging representation works better than the iDF version, so we only report on the latter henceforth.

4.2.2 Preprocessing

We consider for this task the textual content of each talk as represented by two fields in the dataset: *Transcript*, which is automatically generated for each talk, and *Description*, which is a short summary of the content of the talk, as written by an editor.

We consider a simple preprocessing step where we lowercase the text, remove stop words and punctuation, and then we lemmatize all words for all but the contextualized representation (S-BERT). Lemmatization is very important for the tasks of semantic matching, as it allows us to avoid counting plurals or different forms of verbs as different words, which lowers the number of surface forms in some representations, and allows to find a pre-computed embeddings for each word in the text when using Word Embeddings. We also remove stop words and the top frequent words, as they do not particularly contribute to the task of content matching (empirically, we observe that they actively harm it). It is however not necessary for the contextualized representation as the tokenization and representation of plurals and forms is integrated into the transformer pipeline (i.e. S-BERT).

4.3 Semantic Representation and Automatic Annotations

To capture a different level of semantic knowledge about the content of the talks, we explore the use of knowledge graphs and their latent representations.

[5] https://radimrehurek.com/gensim/models/doc2vec.html
[6] https://github.com/UKPLab/sentence-transformers
[7] https://huggingface.co/sentence-transformers/bert-base-nli-stsb-mean-tokens

We describe the process of creating the knowledge graph, as well as the novel enrichment process that we propose to improve said representations.

4.3.1 Building the Knowledge Graph

To create the "semantic representations" for the talks, we start by building a knowledge graph based on the metadata that is available in the dataset. This metadata includes "Event", "Speaker", "Tag" and "Theme". Using these metadata elements, we can build an initial knowledge graph that connects talks which some of these descriptors. This creates the initial KG. Next, we can use a graph embedding method [30] to generate a fixed-dimensional embedding for each talk in the dataset, such that talks having similar annotations would be represented in proximity in the embedding space. As a result, we can measure the (cosine) similarity between any two talks' embeddings as a proxy to their relatedness. For instance: talks that share the same "theme" will be closer together in the embedding space.

In this work, we do not differentiate semantically between nodes that represent the talks and nodes that represent metadata. All nodes are "first-class citizens" in the KG, and it only contains a single predicate: "describes" (pointing from the metadata nodes to the talk nodes). The resulting knowledge graph has the statistics presented in Table 1.

Element	Count
Talks	1149
Unique tags	299
Unique themes	47
Speakers	1006
Events	132
Relation (triplets)	12711

Table 1 Statistics about the created knowledge graph for the TED dataset

4.3.2 Choice of Embeddings

Throughout the experiments section, we generate a graph connecting the talks and their annotations. Next, we compute node embeddings for each talk in our dataset. While this choice may be important for the overall performance of the final recommendation system, it is out of scope for this work. Our focus in this paper is to demonstrate the complementary of textual and semantic representations, and the utility of automatic annotations for improving content recommendation.

To bypass the need to select a proper graph embedding technique and the expensive hyperparameter fine-tuning that goes with it for each experiment, we start from the KG containing the talks and their manually annotated metadata from the original TED dataset, i.e. *tags* and *themes*. This would allow us to create a Knowledge Graph that does not contain any noisy or extraneous

annotations. We compute the node embeddings for each talk using a selection of embedding algorithms contained in the `Pykg2vec` package[8], a Python library for learning representations of entities and relations in Knowledge Graphs using state-of-the-art models [31]. We fine-tune each representation using a small grid-search optimization over learning rate, embedding size and number of training epochs. We also add the one-hot encoding of each talk (each talk is represented by a binary vector which represent the presence or absence of each tag and theme in the metadata) to see if there is an advantage for using graph embeddings over a simple flat representation of the nodes, i.e. whether the graph embeddings encode some semantics between the annotations that a simple binary representation cannot pick up on (e.g. the presence of one tag may be related to some other tag/theme, in other words that the annotations are not mutually orthogonal).

4.3.3 Automatic Annotations

On top of the initial KG (containing only metadata from the dataset), we propose to use several Information Extraction techniques such as Topic Modeling, Named Entity Recognition, and Keyword Extraction, to generate high level descriptors – *annotations* – of the content of each video in the dataset. Once the annotations are generated for each video, we use them to enrich the Knowledge Graph connecting the talks by their annotations. This approach also allows us to integrate external metadata if such metadata is available. For our dataset, metadata such as "Tags" and "Themes" are available and will be used. The approach is illustrated in Figure 1.

For each of the studied automatic annotations, we start by running our automatic annotation model. We then create a Knowledge Graph using on one hand the metadata provided in the dataset (each talk is labeled with a "tag" and a "theme"), and our automatically extracted descriptors on the other hand. Once we connect all the talks using these annotations, we run a Graph Embedding method (see Section 4.3.2) to generate an embedding for each talk in the dataset. These embeddings serve then as representations that we can use to measure similarities for both **T1** and **T2**.

We present a selection of automatic annotations techniques and how they are used in our approach in the following paragraphs.

Topic Modeling

Topic modeling is a ubiquitously used Information Extraction technique, which attempts to find the latent topics in a text corpus. A topic can be roughly defined as a coherent set of vocabulary words that tend to co-appear with high probability in the same documents. When applied on documents of natural language, topic models have the ability to find the underlying "themes" in the document collection, such as sport, technology, etc.

The literature on topic modeling is rich and diverse, with approaches relying solely on word counts such as the commonly used LDA [32], to using

[8] https://github.com/Sujit-O/pykg2vec

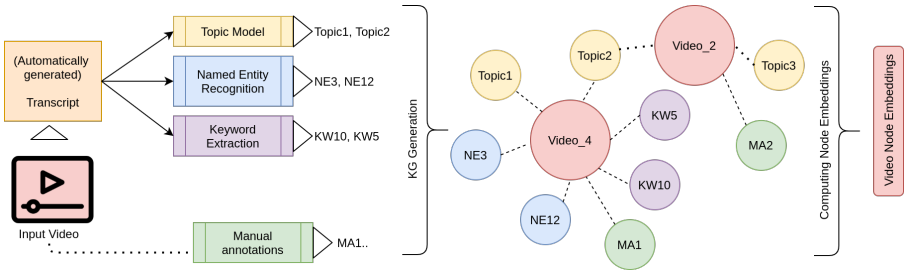

Fig. 1 High level illustration of our approach: we start by extracting annotations from the video transcript using off-the-shelf Information Extraction tools, which we combine with manual annotations to create a Knowledge Graph, where the talks and the annotations are nodes, connected with the corresponding semantic relation. Using this graph structure, we can generate continuous fixed-dimensional representations using a Graph Embedding technique, which we can later use to measure content similarity for recommendation.

state-of-the-art representations to represent documents in more meaningful representational spaces [33, 34]. Topics are usually represented with their "top N words" (the N words most likely to appear given a topic). In our dataset, we find topics such as:

- *Technology*: network,online,computers,digital,google
- *Environment*: waste,plants,electrical,plastic,battery
- *Gaming*: games,online,virtual,gamers,penalty
- *Health*: aids,malaria,drugs,mortality,vaccine

For our experiments, we use *LDA* as it is still commonly used and offers simple yet competitive performance [29]. We test two aspects of topic modeling that can influence the structure of the graph (the number of nodes and relations added) which are: the number of topics (i.e. the number of topic nodes in the final KG), as well as the cutoff threshold reflecting the topic model's confidence is assigning a given topic to a given talk (which would affect the number of relations to topic nodes). We report the results in Section 5.2. For a better performance of the topic modeling task, we pre-process our dataset as follows:

1. Lowercase all words
2. Remove short words (less than 3 characters)
3. Remove punctuation
4. Remove the most frequent words (top 1%)

Named Entity Recognition

Named Entity Recognition is the task of extracting from unstructured text, terms or phrases that refer to named entities, i.e. real world objects that have proper names and can refer to one of several classes: persons, places, organizations, etc. Once extracted, these Named Entities can be used as high level descriptors for a text content. For example, if two talks mention "Einstein" and "Newton", they may have a similar topic. While this task used to rely on grammatical and hand-crafted features to designate what would constitute a

Named Entity (e.g. starts with a capital letter), modern systems do without such hand crafted features [11], but rely on combining the learning power of neural networks with annotated corpora of Named Entities.

In our experiments, we use SpaCy's [35] NER model which uses an architecture that combines a word embedding strategy using sub word features, and a deep convolution neural network with residual connections, which is "designed to give a good balance of efficiency, accuracy and adaptability"[9].

For our experiments, we keep the Named Entities belonging to the following classes: 'PERSON', 'LOC' (location), 'ORG' (organization), 'GPE' (geopolitical entity), 'FAC' (faculty), 'PRODUCT', and 'WORK_OF_ART'. We also experiment with the impact of keeping all extracted Named Entities or filtering some out based on frequency, thus altering the number of added nodes to the graph and their relations to the existing talks. We report the results in Section 5.2.

Keyword Extraction

Similarly to the two previous tasks, Keyword Extraction is the process of extracting terms of phrases that summarize on a high level the core themes of a textual document. Generally, the keywords (or sometimes called tags) are the terms or phrases that are explicitly mentioned in the text with a high frequency or are somehow relevant to a big portion of it.

For our experiments, we use KeyBERT [36], an off-the-shelf keyword extractor that is based on BERT [11], which extracts keywords by first finding the frequent n-grams, then measuring the similarity between their embedding and the embedding of the whole document. We experiment with keeping all keywords or filtering out rare ones and report the results in Section 5.2.

4.4 Combining Representations

Because both textual (Section 4.2) and semantic (Section 4.3) representations rely on generating a vector embedding of the talk, we can combine them in straightforward way by just averaging the similarity scores obtained by both representations, thus ensuring that items that are similar in either or both representation spaces would have a higher combined similarity score. The upshot, hopefully, is that the two embeddings capture different semantics, and thus, such combination, albeit straightforward and simple, improves the results on recommendation metrics.

We also observe that other representation and similarity scores can be added, e.g. a visual embeddings similarity for the video content. Because the TED talks are not visually diverse, they do not offer much in the visual modality to derive interesting similarity measure, but this could be an interesting modality for more challenging datasets.

[9] urlhttps://spacy.io/universe/project/video-spacys-ner-model

5 Experiments and Results

In this section, we describe the results for the different experiments done to study the impact of using the different representations described above and the effect of adding automatic annotations on recommendation performance. We start by showing the results on different textual representations. Then, we do the same for the different graph embeddings. After that, and for each automatic annotation considered (i.e. Topics, Named Entities and Keywords), we consider several configurations, with and without the addition of the original metadata from the dataset. Finally, we observe the potential of combining the resulting automatically generated graph embeddings with the textual embeddings of the content, and show how the two complement each other to push the empirical performance even higher.

5.1 Performance of Textual Representations

As explained in Section 3.1, we evaluate the performance of the aforementioned textual embeddings by computing an embedding for each talk, then, for T1, an embedding for each user by averaging the items in their favorites history (minus the left-out item for the test set). In all of the following experiments, we use bootstrapping [37] to test the performance of our representations. We report on the mean and standard deviation (in parenthesis) based on 10 re-samples of the evaluation set.

In Table 2, we compare the performance of several textual representations on the T1 recommendation task. The table also shows the impact of using the human-created description field (which is written by editors) to the use of transcripts, which can be automatically generated and thus be used at scale.

As we can see from the results, the simple TF-iDF representation performs the best at this task. This is probably due to the fact that exact matching of terms is more important than the soft matching that other unsupervised textual representations allow (topics, embeddings). The particularly poor performance of Doc2Vec may be due to the relatively small size of the dataset, making the self-supervised approach generate poorer representations.

It is also worth noting that using the transcripts as representation of the content of the talk yields comparable results to using the handwritten description of the video (the difference in performance is statistically insignificant), suggesting that the transcripts that are extracted using automatic speech recognition (ASR) can be used as good off-the-shelf representations for the textual content of spoken media (i.e. the TED talks here), and the human input (in the form of summaries) is not necessary for this task. Combining the two yields only marginal improvement on some of the representations, suggesting that the information contained in the summary probably contains redundant information.

In Table 3, we perform the same experiments for the task T2, the general content-based recommendations. Again, we notice several similar patterns in the results: the performance of either *Description* or *Subtitles* are on par for

Representation	HIT@10	MRR@10
	Description	
TF-iDF	**0.0695** (0.0025)	**0.0304** (0.0010)
NMF	0.0302 (0.0011)	0.0109 (0.0006)
GloVe	0.0550 (0.0017)	0.0232 (0.0008)
Doc2Vec	0.0086 (0.0009)	0.0024 (0.0004)
FastText	0.0513 (0.0021)	0.0185 (0.0009)
S-BERT	0.0476 (0.0021)	0.0212 (0.0010)
	Transcript	
TF-iDF	**0.0694** (0.0029)	**0.0304** (0.0016)
NMF	0.0291 (0.0030)	0.0109 (0.0012)
GloVe	0.0540 (0.0036)	0.0231 (0.0016)
Doc2Vec	0.0091 (0.0005)	0.0025 (0.0003)
FastText	0.0516 (0.0028)	0.0190 (0.0018)
S-BERT	0.0473 (0.0023)	0.0212 (0.0010)
	Transcript + Description	
TF-iDF	**0.0696** (0.00314)	**0.0300 (0.0012)**
NMF	0.0294 (0.0020)	0.0106 (0.0010)
GloVe	0.0536 (0.0021)	0.0223 (0.0009)
Doc2Vec	0.0093 (0.0011)	0.0026 (0.0004)
FastText	0.0518 (0.0027)	0.0191 (0.0014)
S-BERT	0.0467 (0.0020)	0.0207 (0.0012)

Table 2 The performance of several textual representations (using talks description and transcript) on **T1**

most representations, TF-iDF still performs the best overall, and combining both the description and the transcripts give slightly better results for almost all representations.

Overall, we find that for both **T1** and **T2**, using simple off-the-shelf textual representations can be useful for content-based recommendation, with the simpler combination of TF-iDF and lemmatization in preprocessing giving the best performance in both cases, which may suggest that at the document (talk) level, the semantic information is not well captured by the off-the-shelf models, and the low-level similarity (word matching via TF-iDF) works best. On a more interesting note, it seems that whatever works for **T1** equally works for **T2**, suggesting a correlation between the efficiency of representations for both user preferences and generic content-based recommendations, and that the research done on one task can be then transposed to the other, and vice-versa.

5.2 Semantic Representations

We report the results using semantic representations in Tables 4 and 5, for **T1** and **T2**, respectively.

From these tables of results, we make the following observations:

Representation	HIT@10	MRR@10
	Description	
TF-iDF	0.2410 (0.0076)	0.1685 (0.0035)
NMF	0.0973 (0.0030)	0.0903 (0.0026)
GloVe	0.2396 (0.0072)	0.1834 (0.0051)
Doc2Vec	0.0094 (0.0021)	0.0049 (0.0012)
FastText	**0.2514** (0.0109)	**0.1903** (0.0076)
S-BERT	0.2267 (0.0055)	0.1674 (0.0048)
	Transcript	
TF-iDF	**0.3047** (0.0099)	**0.1996** (0.0083)
NMF	0.0291 (0.0958)	0.0572 (0.0026)
GloVe	0.2439 (0.0047)	0.1528 (0.0050)
Doc2Vec	0.0141 (0.0025)	0.0061 (0.0016)
FastText	0.2325 (0.0088)	0.1474 (0.0048)
S-BERT	0.0928 (0.0059)	0.0593 (0.0028)
	Transcript + Description	
TF-iDF	**0.3319** (0.0048)	**0.2224** (0.0042)
NMF	0.1286 (0.0043)	0.0809 (0.0042)
GloVe	0.2574 (0.0082)	0.1692 (0.0051)
Doc2Vec	0.0148 (0.0028)	0.0055 (0.0006)
FastText	0.2414 (0.0109)	0.1803 (0.0076)
S-BERT	0.2133 (0.0085)	0.1561 (0.0075)

Table 3 The performance of several textual representations (using talks description and transcript) on **T2**

Embedding method	HIT@10	MRR@10
ConvE	0.0162 (0.0014)	0.0058 (0.0010)
DistMult	0.0094 (0.0013)	0.0023 (0.0003)
NTN	0.0489 (0.0030)	0.0184 (0.0011)
Rescal	0.0113 (0.0007)	0.0036 (0.0003)
TransD	**0.0793** (0.0021)	**0.0311** (0.0011)
TransE	0.0596 (0.0019)	0.0221 (0.0009)
TransH	0.0598 (0.0027)	0.0230 (0.0011)
TransM	0.0674 (0.0028)	0.0256 (0.0012)
TransR	0.0607 (0.0022)	0.0220 (0.0011)
One-hot	0.0591 (0.0027)	0.0218 (0.0013)

Table 4 The performance of different graph embedding methods on T1

– Over the studied configurations of hyperparameters, models generally have the same ranking in performance whether used on **T1** or **T2**, i.e. models which perform well on one task tend to perform well on the other task. This means that whatever properties an embedding method has, they seem to translate similarly on both tasks. The poor performance of some methods may be due to their high sensitivity to hyperparameter fine tuning.
– Over the studied configurations of hyperparameters, translation-based methods perform the best empirically, with `TransD` [38] performing the best (by

Embedding method	HIT@10	MRR@10
ConvE	0.0174 (0.0014)	0.0089 (0.0065)
DistMult	0.0073 (0.0012)	0.0034 (0.0007)
NTN	0.1238 (0.0086)	0.0715 (0.0035)
Rescal	0.0086 (0.0016)	0.0044 (0.0011)
TransD	**0.2508** (0.0070)	**0.1587** (0.0065)
TransE	0.2103 (0.0093)	0.1280 (0.0076)
TransH	0.2120 (0.0079)	0.1353 (0.0071)
TransM	0.2246 (0.0073)	0.1337 (0.0064)
TransR	0.1934 (0.0075)	0.1136 (0.0057)
One-hot	0.2204 (0.0095)	0.1216 (0.0077)

Table 5 The best performance of different graph embedding methods on T2

quite a margin) in both set of experiments. While further experiments may be needed to determine how much this performance is due to the nature of the dataset (size, sparsity, etc.) and the task itself, for our experiments, we will take this model as our embedding method of choice (with a learning rate of 0.001, embedding and hidden size of 300, all trained for 1000 epochs, the other hyperparameters being left at their default values).

– One-hot node embeddings perform well on both tasks, which shows that on clean, controlled, human-annotated metadata, a simple exact matching of metadata is good enough to produce good results. The fact that `TransD` outperforms One-hot embeddings even in this setting shows that the graph embeddings capture some semantics beyond exact matching, which means that it learns to find latent meaning between the tags and themes, which ultimately justifies the use of graph embeddings.

5.3 Automatic Annotations

In this section, we observe the performance gain of the different automatic annotations methods we have introduced in Section 4.3.3.

5.3.1 Topic Modeling

In Table 6, we report on the results of adding the output of the topic modeling annotations to the KG. We evaluate the results as we vary two parameters: the number of topics and the cutoff threshold (the confidence score above which we assign a talk to a given topic).

From this small sample of hyperparameters values, we see that both the number of topics and the cutoff threshold impact the performance of the recommendation on both tasks. Performance improves when raising the cutoff threshold, which implies that when we only assign topics to talks, if the topic model is highly confident, it decreases the noisy relations in the graph and decrease the risk of accidentally connecting nodes that are not really topically similar. We also note that under the right configuration, we improve the performance on both metrics for both tasks, whereas in most other configurations the performance suffers. We note that with the number of topics one should

# topics	Threshold	HIT@10	MRR@10
		T1	
No topics added		0.0793 (0.0021)	0.0311 (0.0011)
10	0.03	0.0722 (0.0022)	0.0276 (0.0012)
10	0.3	0.0729 (0.0019)	0.0282 (0.0011)
40	0.03	0.0809 (0.0023)	0.0317 (0.0016)
40	**0.3**	**0.0812** (0.0028)	**0.0337** (0.0018)
100	0.03	0.0597 (0.0017)	0.0223 (0.0017)
100	0.3	0.0628 (0.0017)	0.0243 (0.0018)
		T2	
No topics added		0.2508 (0.0070)	0.1587 (0.0065)
10	0.03	0.2187 (0.0083)	0.0343 (0.0056)
10	0.3	0.2247 (0.0076)	0.1312 (0.0069)
40	0.03	0.2488 (0.0096)	0.1623 (0.0071)
40	**0.3**	**0.2566** (0.0081)	**0.17133** (0.0072)
100	0.03	0.2047 (0.0076)	0.1261 (0.0073)
100	0.3	0.2161 (0.0080)	0.1302 (0.0081)

Table 6 The results of enriching the metadata KG with Topic nodes, varying the number of topics and the cutoff threshold

find a value that is befitting the studied corpus, as the value 40 (inspired by the ground truth number of *themes* in the dataset) seems to give the best results.

Topic modeling is a task that is generally very sensitive to the initial hyper-parameters and subject to inherent stochasticity, which means that with enough experiments, it is likely to find a configuration of hyperpamaters (not only the number of topics and the cutoff threshold but also model-specific hyperparameters such as LDA's *alpha* and *beta*) that yields even better improvement over the reported results.

5.3.2 Named Entity Recognition

In Table 7, we report on the results of adding the output of the Named Entity Recognition annotations to the KG. We evaluate the results as we switch between keeping all entities we extracted in the KG and keeping only ones that appear with a high enough frequency: in our case, we only add nodes for entities that are mentioned more than 10 times in the corpus.

From these results, we see that adding NEs improves the results of the recommender system, especially after removing rarely appearing Named Entities (either erroneous or superfluous mentions). We also notice that MRR increases significantly with this addition for **T2**, suggesting that the Named Entities are strong indicators of content relatedness.

5.3.3 Keywords Extraction

In Table 8, we report on the results of adding the output of the Keyword Extraction to the KG. We evaluate the results as we add either all extracted

# mentions	HIT@10	MRR@10
	T1	
No NEs added	0.0793 (0.0021)	0.0311 (0.0011)
All NEs added	0.0802 (0.0028)	0.0310 (0.0015)
More than 10 mentions	**0.0833** (0.0030)	**0.0311** (0.0012)
	T2	
No NEs added	0.2508 (0.0070)	0.1587 (0.0065)
All NEs added	0.2499 (0.0075)	0.1590 (0.0072)
More than 10 mentions	**0.2620** (0.0069)	**0.1823** (0.0071)

Table 7 The results of enriching the metadata KG with Named Entity nodes, varying the number of filtered entities

keywords or only the ones that the keyword extraction model assigned a high enough confidence score to. In our experiment, a confidence score above 0.3 has been chosen.

Confidence	HIT@10	MRR@10
	T1	
No KWs added	0.0793 (0.0021)	0.0311 (0.0011)
All KWs added	0.0758 (0.0022)	0.0290 (0.0011)
Only with conf $>$ 0.3	**0.0802** (0.0023)	**0.0328** (0.0013)
	T2	
No KWs added	0.2508 (0.0070)	0.1587 (0.0065)
All KWs added	0.2412 (0.0068)	0.1566 (0.0067)
Only with conf $>$ 0.3	**0.2539** (0.0078)	**0.1610** (0.0070)

Table 8 The results of enriching the metadata KG with Keywords nodes, varying the confidence threshold

5.3.4 Combining All Annotations

In Table 9, we summarize the results from previous experiments, and we see that the addition of the best configuration from each experimental setting into one KG further improves the results.

We observe that the automatic annotations overall improve the performance on the recommendation task on purely content-based recommendations (T2, a relative Hit Rate improvement of 8.7%), but surprisingly, they do so even for user preference-based ones (T1, a relative Hit Rate improvement of 11.%), although the overall performance is still significantly lower. One could argue that this is because users are usually interested in similar content to what they watched previously (in other words, all recommendation tasks are partially content-based). There is a possibility, however, that the user is likely

Annotation	HIT@10	MRR@10
	T1	
No annotations added	0.0793 (0.0021)	0.0311 (0.0011)
Topics	0.0812 (0.0028)	0.0336 (0.0018)
Named Entities	0.0833 (0.0030)	0.0311 (0.0012)
Keywords	0.0802 (0.0023)	0.0328 (0.0013)
All	**0.0916** (0.0031)	**0.0439** (0.0015)
	T2	
No annotations added	0.2508 (0.0070)	0.1587 (0.0065)
Topics	0.2566 (0.0081)	0.1713 (0.0072)
Named Entities	0.2620 (0.0069)	**0.1823** (0.0071)
Keywords	0.2539 (0.0078)	0.1610 (0.0070)
All	**0.2749** (0.0087)	0.1761 (0.0059)

Table 9 The results on both recommendation tasks with all the different annotations added to the KG

to click on the suggested video in the "related" section, which creates a dependence between the two tasks that is impossible to untangle. This is beyond the scope of this paper, but it is interesting to study the feedback loop of recommendation in such setting. Finally, the results suggest that Named Entity Recognition contributes the most to the overall performance improvement of the system, as it is the closest to the overall performance and still gives a better absolute MRR score.

5.4 Combining Semantic and Linguistic Representations

In this section, we build up on the results obtained in Sections 4.2 (textual) and 4.3 (semantic) to further improve the results of recommendations. Table 10 shows the performance gain upon combining the semantic and linguistic representations on both recommendations tasks.

Representation	HIT@10	MRR@10
	T1	
TransD on KG	0.0916 (0.0031)	**0.0439** (0.0015)
TF-iDF on Transcript	0.0694 (0.0029)	0.0304 (0.0016)
Combined	**0.1005** (0.0025)	0.0419 (0.0016)
	T2	
TransD on KG	0.2749 (0.0087)	0.1761 (0.0059)
TF-iDF on Transcript	0.3047 (0.0099)	0.1996 (0.0083)
Combined	**0.3306** (0.0055)	**0.2375** (0.0045)

Table 10 The performance improvement by combining semantic and linguistic representations

From the results on both recommendation tasks, we see that even the simple scheme of averaging similarity scores from the two different modalities lead generally of significant improvement on both metrics. Even though there is a noticeable difference between the modalities (the semantic representation outperforms the linguistic one on T1, and the inverse is shown for T2), averaging the two scores did not net a Hit Rate/MRR that is the average of the two individual scores, but a significantly higher one (16.8% and 10% relative Hit Rate improvement w.r.t the best modality on T1 and T2, respectively). This clearly suggests that the two representations are *complementary*, i.e. whatever is captured in one representation is not necessarily covered in the other, even though they are both based on the talk content. Thus, combining the two similarity scores make the overall recommender system better.

These results not only confirm that the combination of different representation spaces and methods is a simple and basically free way of improving the recommendation task (both user-based and content-based), but as it is shown that even with a simple linear combination of similarity score, an immediate and significant improvement of the results can be obtained, they also suggests a interesting line of research on how to combine the different representations and how specific combinations can quantitatively and qualitatively alter the nature of recommendations made by the system.

6 Conclusion and Future Work

In this work, we showed how combining the textual representation of media content with the semantic representation obtained by extracting knowledge automatically using Information Extraction techniques can improve the performance of content-based media Recommender Systems without requiring any supervision or external data collection, as we demonstrated clear performance improvement measured on two tasks: recommendations based on manually curated related items, and recommendations based on actual users interaction history. The empirical results suggest that the two representations capture different levels of similarity: low-level "word matching" and high-level "semantics" through the KG embeddings. Our results are reproducible using the code published at https://github.com/D2KLab/ka-recsys.

With these promising results, there are multiple paths for further exploration. On the linguistic side, several other representations can be tried out, and a combination of multiple representations can lead to more robust similarity assessment, as count-based, distributional and neural representations tend to capture different aspects of the "meaning" of the content and thus can be complementary.

On the semantic side, other techniques from the information extraction literature can be investigated such as entity linking, aspect extraction, concept mining, as well as information extracted from other modalities (e.g. visual or audio). What is more, as shown experimentally, the way these automatic annotations are processed and filtered (thus changing the structure of the

generated KG), the results can vary, which calls for further study of how to balance the quantity of automatic annotations and the cutback on the necessary noise that comes with it. Another direction of work is to further explore models that go beyond simple graph embeddings. Furthermore, as these extracted annotations live on a KG, multiple methods in the direction of *Explainable Recommendations* can be explored in tandem.

Acknowledgment

This work has been partially supported by CHIST-ERA within the CIMPLE project (CHIST-ERA-19-XAI-003) and by raisin.ai within the MyLittleEngine project.

References

[1] Kotkov, D., Wang, S., Veijalainen, J.: A survey of serendipity in recommender systems. Knowledge-Based Systems **111**, 180–192 (2016)

[2] Kunaver, M., Požrl, T.: Diversity in recommender systems – a survey. Knowledge-Based Systems **123**, 154–162 (2017)

[3] Zhang, Y., Chen, X.: Explainable recommendation: A survey and new perspectives. Found. Trends Inf. Retr. **14**, 1–101 (2020)

[4] Pappas, N., Popescu-Belis, A.: Combining content with user preferences for ted lecture recommendation. In: 11th International Workshop on Content-Based Multimedia Indexing (CBMI), pp. 47–52 (2013)

[5] Mishra, A., Vishwakarma, S.: Analysis of tf-idf model and its variant for document retrieval. In: 2015 International Conference on Computational Intelligence and Communication Networks (CICN), pp. 772–776 (2015)

[6] Yi, X., Allan, J.: A comparative study of utilizing topic models for information retrieval. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) Advances in Information Retrieval, pp. 29–41. Springer, Berlin, Heidelberg (2009)

[7] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)

[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)

[9] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors

with subword information. arXiv preprint arXiv:1607.04606 (2016)

[10] Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: 31st International Conference on International Conference on Machine Learning (ICML), pp. 1188–1196. JMLR.org, ??? (2014)

[11] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL, Minneapolis, USA (2019)

[12] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: EMNLP-IJCNLP 2019, pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (2019)

[13] Lops, P., Jannach, D., Musto, C., Bogers, T., Koolen, M.: Trends in content-based recommendation: Preface to the special issue on recommender systems based on rich item descriptions. User Modeling and User-Adapted Interaction **29** (2019). https://doi.org/10.1007/s11257-019-09231-w

[14] Chicaiza, J., Valdiviezo-Diaz, P.: A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions. Information **12**(6) (2021). https://doi.org/10.3390/info12060232

[15] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc., ??? (2013)

[16] Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.-Y.: Collaborative knowledge base embedding for recommender systems. In: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 353–362. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939673. https://doi.org/10.1145/2939672.2939673

[17] de Gemmis, M., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Semantics-Aware Content-Based Recommender Systems, pp. 119–159. Springer, Boston, MA (2015)

[18] Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM **38**(11), 39–41 (1995)

[19] Polignano, M., Musto, C., de Gemmis, M., Lops, P., Semeraro, G.:

Together is Better: Hybrid Recommendations Combining Graph Embeddings and Contextualized Word Representations, pp. 187–198. Association for Computing Machinery, New York, NY, USA (2021)

[20] Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., He, Q.: A Survey on Knowledge Graph-Based Recommender Systems. arXiv (2020). https://arxiv.org/abs/2003.00911

[21] Cao, Y., Wang, X., He, X., Hu, Z., Tat-seng, C.: Unifying knowledge graph learning and recommendation: Towards a better understanding of user preference. In: World Wide Web Conference (WWW) (2019)

[22] Harrando, I., Troncy, R.: Improving media content recommendation with automatic annotations. In: 3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) & 5th Edition of Recommendation in Complex Environments (ComplexRec) Joint Workshop @ RecSys 2021. CEUR Workshop Proceedings, Amsterdam, Netherlands (2021). http://ceur-ws.org/Vol-2960/paper16.pdf

[23] Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems, pp. 291–324. Springer, Berlin, Heidelberg (2007)

[24] Pappas, N., Popescu-Belis, A.: Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In: ACM SIGIR 2013, pp. 773–776 (2013)

[25] Merchant, A., Singh, N.: Hybrid trust-aware model for personalized top-n recommendation. In: Fourth ACM IKDD Conferences on Data Sciences. Association for Computing Machinery, Chennai, India (2017)

[26] Pappas, N., Popescu-Belis, A.: Combining content with user preferences for non-fiction multimedia recommendation: a study on ted lectures. Multimedia Tools and Applications **74**, 1175–1197 (2013)

[27] Sun, R., Cao, X., Zhao, Y., Wan, J., Zhou, K., Zhang, F., Wang, Z., Zheng, K.: Multi-modal knowledge graphs for recommender systems. In: 29th ACM International Conference on Information and Knowledge Management (CIKM), pp. 1405–1414. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3411947

[28] Rendle, S.: Factorization machines. In: IEEE International Conference on Data Mining, pp. 995–1000 (2010)

[29] Harrando, I., Lisena, P., Troncy, R.: Apples to apples: A systematic evaluation of topic models. In: RANLP, vol. 260, pp. 488–498 (2021)

[30] Cai, H., Zheng, V., Chang, K.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering **30**, 1616–1637 (2018)

[31] Yu, S.Y., Rokka Chhetri, S., Canedo, A., Goyal, P., Faruque, M.A.A.: Pykg2vec: A Python Library for Knowledge Graph Embedding. arXiv (2019)

[32] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation **3**, 993–1022 (2003)

[33] Bianchi, F., Terragni, S., Hovy, D.: Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. In: ACL-IJCNLP 2021, pp. 759–766. Association for Computational Linguistics, Online (2021)

[34] Tian, T., Fang, Z.F.: Attention-based autoencoder topic model for short texts. Procedia Computer Science **151**, 1134–1139 (2019). ANT 2019 / EDI40 2019

[35] Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python. Zenodo (2020). https://doi.org/10.5281/zenodo.1212303

[36] Grootendorst, M.: KeyBERT: Minimal keyword extraction with BERT. Zenodo (2020). https://doi.org/10.5281/zenodo.4461265

[37] Efron, B.: Bootstrap Methods: Another Look at the Jackknife. The Annals of Statistics **7**(1), 1–26 (1979)

[38] Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: ACL (2015)