Journal of Intelligent & Fuzzy Systems 34 (2018) 3089–3100
DOI:10.3233/JIFS-169493
IOS Press

Global citation recommendation using knowledge graphs

Frederick Ayala-Gómez[a,*], Bálint Daróczy[b], András Benczúr[b], Michael Mathioudakis[c] and Aristides Gionis[d]
[a]*Eötvös Loránd University, Faculty of Informatics, Budapest, Hungary*
[b]*Inst. Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary*
[c]*Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, France*
[d]*Department of Computer Science, Aalto University, Espoo, Finland*

Abstract. Scholarly search engines, reference management tools, and academic social networks enable modern researchers to organize their scientific libraries. Moreover, they often provide recommendations for scientific publications that might be of interest to researchers. Because of the exponentially increasing volume of publications, effective citation recommendation is of great importance to researchers, as it reduces the time and effort spent on retrieving, understanding, and selecting research papers. In this context, we address the problem of *citation recommendation*, i.e., the task of recommending citations for a new paper. Current research investigates this task in different settings, including cases where rich user metadata is available (e.g., user profile, publications, citations). This work focus on a setting where the user provides only the abstract of a new paper as input. Our proposed approach is to expand the semantic features of the given abstract using knowledge graphs – and, combine them with other features (e.g., indegree, recency) to fit a learning to rank model. This model is used to generate the citation recommendations. By evaluating on real data, we show that the expanded semantic features lead to improving the quality of the recommendations measured by nDCG@10.

Keywords: Citation recommendations, knowledge graphs, recommender systems

1. Introduction

Finding relevant bibliography is time-consuming for modern researchers – and it is easy to miss existing related work due to the large volume of scientific literature. Over recent decades, the number of scientific articles has exhibited exponential increase, and this trend is not expected to change soon [1, 2]. Lately, new publication channels such as conference proceedings and open archives have added to the volume of publications and are also growing at a fast pace. This growth is exciting but challenging, as it increases the time that researchers must spend on finding relevant papers to their specific research projects.

To help with this problem, several digital tools such as reference management software, digital libraries, and search engines offer research paper recommendations to their users. Following [3], we highlight two types of research paper recommendations. The first is *Global Recommendations*, where the task is to find relevant articles related to a given corpus as a whole. The second is *Local Recommendations*, where the task is to detect *local contexts* (e.g., sentences, paragraphs in a paper) for which there exist related papers in the literature.

Recommender systems have found application in many different domains, including movies, music, videos, news, books, and products in general.

*Corresponding author. Frederick Ayala-Gómez, Eötvös Loránd University, Faculty of Informatics, 1117 Budapest, Hungary. E-mail: fayala@caesar.elte.hu.

Depending on the context, they produce a list of recommended items using a variety of techniques (e.g., collaborative or content-based filtering [4]). The literature in the field of research paper recommendation is extensive, and content-based recommendations are the most common solution to the global citation recommendation task [5].

In this paper, we research a particular variant of the global recommendations task: we aim to *find papers that are relevant to a short description (abstract) of a new paper, provided as input*. Addressing this task will be beneficial to researchers who find themselves working on a new paper, with an accurate idea about the problem they wish to solve and their approach, but little knowledge of related work in the literature. For such cases, which are very common among professional researchers, we envision recommender systems that allow the researchers to put together a short text (an abstract) that describes the problem and approach they consider, submit it to the system, and receive a set of related papers.

An essential contribution of our proposed approach is to add semantic features by mapping the given abstract to knowledge graphs. Knowledge graphs represent structured and detailed knowledge about a topic via labeled-nodes and labeled-edges, usually following the Resource Description Framework (RDF)[1]. One of the most extensive knowledge graphs is DBpedia [6]. It contains topics and facts from Wikipedia, 'the largest and most popular general reference work on the Internet'[2]. We map the terms in the abstract to DBpedia using DBpedia Spotlight [7]. In doing so, we generate two types of semantic features: *Set of Entities*, which contains DBpedia entities, and *Set of Properties*, which contains the set of properties for the mapped entities.

Our approach for producing recommendations for a given abstract has two phases: *training* and *recommendation*. The goal of the training phase is to fit a 'learning to rank' model on a ground-truth dataset compiled from CiteSeerX and the Microsoft Academic Graph, where we have the paper's abstract and its citation. The recommendation phase takes the abstract of a new paper as input and builds recommendations based on the previously fit model. For both training and recommending, we generate a candidate set for the given abstract using *Lucene's More Like This*[3] on the relevant *terms* and *semantic* features of the abstract. We evaluate our recommendations by holding a testing dataset from the ground-truth data. For each of the abstracts in the testing set, we build a Top-10 recommendation and compute nDCG@10 to measure that relevant (cited) papers appear higher in the rank.

Expanding the semantic features with knowledge graphs, using Lucene for candidate generation, and learning to rank on abstract-candidate features, distinguish our work from existing approaches.

The rest of the paper is organized as follows. Section 2 presents a summary of the related research work in the *global* citation recommendation task. Section 3 presents the working dataset, data cleaning, and exploration, our setting for building Top K Recommendations. Section 4 explains the evaluation methodology, data split, metrics, and baselines. Section 5 presents the experimental results, followed by discussions, and conclusions in Section 6.

2. Related work

The literature on research-paper recommender systems is extensive Ref. [5] presents a recent survey on the field. The authors summarize different approaches in the literature and open problems for this task. Close to our work are the content-based filtering methods. 70% of the surveyed methods used *tf-idf* [8] as weighting scheme for terms. Other approaches consider different aspects of the paper's content. For instance, authors in [3] define context vectors (e.g., weighted terms in the abstract) and score candidate papers by computing the average dot product of these context vectors. In a follow-up work [9], the authors use *Cite-PLSA-LDA* [10]. This approach builds an LDA model from the CiteSeerX [11] word-topic and topic-citation distributions. Another content-based recommender is using *keyqueries* to represent the body of the paper corpus. These *keyqueries* are then used to get similar papers [12].

Enriching the content of the paper is done in [13] with a hybrid recommender system called *Quickstep*. The authors combine collaborative filtering with content. As a first step, they classify the topic of the paper using multi-class classifiers. Then, a profiling algorithm computes the user interests to the topic from the predicted paper topics and the user's browsing logs. The final recommendation is the product of the topic classification confidence and the user topic interest. The authors extend the topics with the taxonomy of

[1] https://www.w3.org/RDF/

[2] https://en.wikipedia.org/wiki/Wikipedia

[3] https://lucene.apache.org/

computer science in the *dmoz* ontology[4]. Combining semantic databases to enrich the user interface is proposed in [14], where the authors design a system that tags an input corpus based on a semantic database defined by the user. These tags are then used to build a user interface that allows exploring related papers based on how similar their text is. In the *Docear* research paper recommender system [15], *mind maps* are used to enable users to navigate research recommendations based on content-based filtering methods.

Graph based research paper recommender systems build a graph and then compute research paper rankings to find the most central or popular papers either using a citation graph [16–18] or using the author's graph [19–21]. The authors in [3, 22] present evidence that *Katz* is a good centrality measure for finding the global relevance of scientific papers. However, [23] mentions that using the citations count (i.e., indegree) is good enough.

Current research on using knowledge graphs to expand the semantic features of the research papers is scarce. Authors in [24] research the extraction of semantic entities from the text using *Wikifier* [25, 26]. Their results show that *Wikifier* annotates general terms rather than domain specific terms and they did not research the recommendation task. The usage of knowledge graphs for building Top K recommendations is presented in *SPrank* [27]. SPrank builds features that represent the connection between users and items using paths between the items and the knowledge graph. Their results show that *SPrank* outperforms popularity and biased Matrix Factorization [28] baselines while SLIM [29] reached similar performance. Authors in [30] propose a model that computes pairwise features between users and papers, and then train a learning to rank model. This model learns the user preference over the candidate papers. Their experiments show that learning to rank models help in the paper recommendation task when user profiles and explicit feedback (i.e., users mark papers as relevant) is known.

Our work is complementary to the previously mentioned literature in several aspects. Our recommender system aims to satisfy the information needs expressed in a given abstract only. That is, we do not require building a profile for the user for recommending papers. This strict requirement forces us to leverage new methods for expanding the semantic features of the given abstract. We research the added value of knowledge graphs used as content to build global citation recommendations and propose an approach that uses this new set of features. In particular, we complement current literature as follows. First, by using *DBpedia Spotlight*, we map the terms in the text to DBpedia, a knowledge graph. This process identifies DBpedia entities URIs, and from DBpedia, we extract the entities properties. Second, to reduce the complexity of training and recommending we propose generating a candidate set using *Lucene's More Like This* method. This method uses the terms in the abstract and the DBpedia entities to query for relevant papers. By doing this, we show that the quality of the candidate set improves. Third, our proposed approach builds abstract-candidate pairwise features from a ground-truth dataset and mark the pairs with candidates cited as relevant. We input these pairwise features to LambdaMART [31], a learning to rank model that uses the pair-wise features to predict the labeled relevance.

To measure the quality of our proposed approach, we apply our proposed solution year-by-year evaluating on the next year. For each paper published in the year following the training, we take the abstract and calculate the candidate set, compute the abstract-candidate pairwise features, and apply the fit LambdaMART model to rank the pairs according to the predicted relevance. Then, we measure the quality of the ranking (i.e., if the candidates are on the top) by computing nDCG@10. Our results show that the proposed approach outperforms traditional text-based baselines and semantic features improve recommendation quality. The code used for data collection, analysis, and experimentation is available for academic purposes[5].

3. Setting

To show how we build the research-paper recommendations we start by presenting the working dataset, the data cleaning steps and an exploration on the data characteristics. Then, we present our approach for generating a candidate set and discuss how we build the training dataset for the learning-to-rank task. Finally, we provide our definitions and notations, and we present the prediction task. Figure 1 presents an overview of our approach for building Top K recommendations.

[4] http://www.dmoz.org/

[5] https://github.com/frederickayala/global_citation_recsys

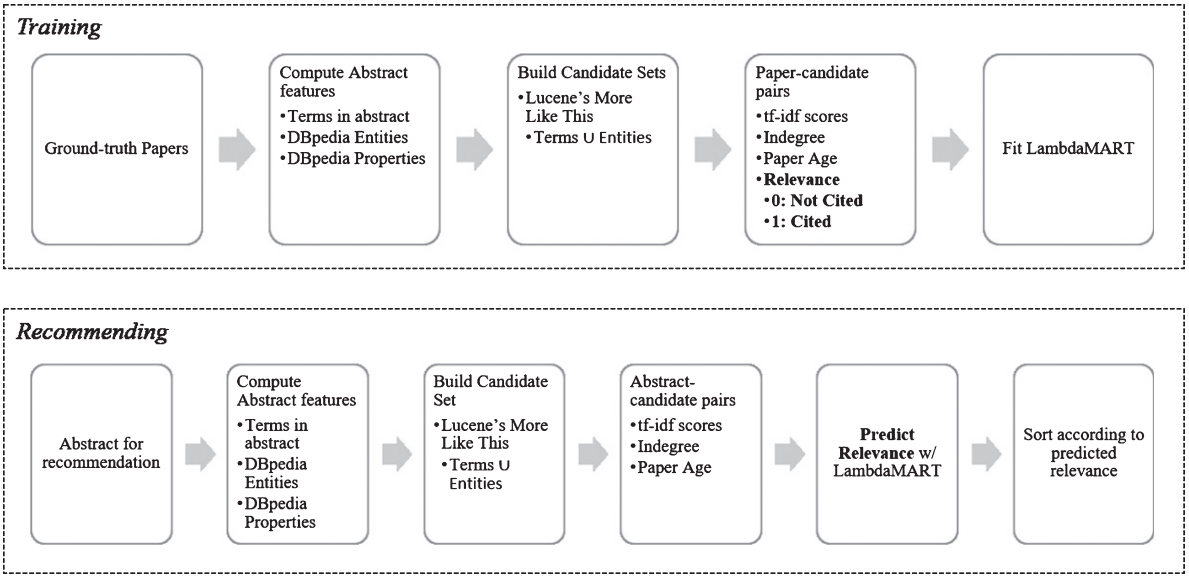

Fig. 1. An overview of our proposed approach for building global citation recommendations. The input of the training phase (top), is the ground-truth dataset, and the output is a LambdaMART model fit to predict the candidate's relevance for the paper. For building recommendations (bottom), we require an abstract that is used to compute content features, build a candidate set, calculate the paper-candidate features, and finally, the trained LambdaMART model is used to predict the relevance of the candidates for the abstract. The recommendations are the set of candidate papers sorted by the predicted relevance.

3.1. Working dataset

We compile a working dataset (i.e., ground-truth dataset) of papers, on which we train a model to predict citations[6] and generate recommendations. We proceed in two steps. In the first step, we combine data from two large, publicly available datasets. The first is CiteSeerX [11], an open-access repository of about 10 million papers. From this dataset, we use the title and abstract – however, the reference lists (citations) are not included[7]. The second dataset is the Microsoft Academic Graph (MAG) [32], a publicly available set of 127 million scientific papers, related metadata, and the list of its references (i.e., cited papers). To merge the CiteSeerX and MAG datasets, we match the papers by the edit distance of their lower-case title[8]. This merge leads to a collection of approximately 2.2 million research papers, 6.7 million authors, and 24 million citations. However, we consider only citations to papers in our dataset. These are 4.7 million citations. Finally, we clean this data as described in Subsection 3.2.

In the second step, we extract semantic features for each paper in the collection. Towards this end, we use DBpedia [6], a large knowledge graph that represents information contained in Wikipedia. Specifically, DBpedia contains facts about entities, expressed as RDF triplets of the form *(subject, predicate, object)* – for example, *(Brooklyn, is_in_city , New York City)*. For each of the papers output from the first step above, we identify the set of RDF *subjects* that are mentioned within its title and abstract – and refer to them as the paper's **entities**. Towards this end, we use DBpedia Spotlight [7], a tested and publicly available tool that annotates input text with DBpedia entities. Moreover, for each paper we also collect the union of RDF objects associated with its Entities - and refer to them as its **properties**. An example of the links between papers, entities, and properties is shown in Fig. 2.

In total, we extracted approximately 270 thousand distinct entities, which account for 25 million annotations, and 1.7 million distinct properties.

To summarize, each paper in the resulting working dataset is associated with the following features: title, abstract, set of citations, set of entities, and set of properties.

3.2. Data cleaning and exploration

In Tables 1 and 2 we present descriptive statistics of the working dataset. We include a cleaning phase

[6]In what follows, citations are the entries in the *Bibliography* of a paper. That is, those papers used as supporting literature. We use the word *citation* and *reference* equally.

[7]The data was collected from *CiteSeerx OAI collection* endpoint, which does not provide the citations.

[8]We used the *FuzzyWuzzy* python module [33].

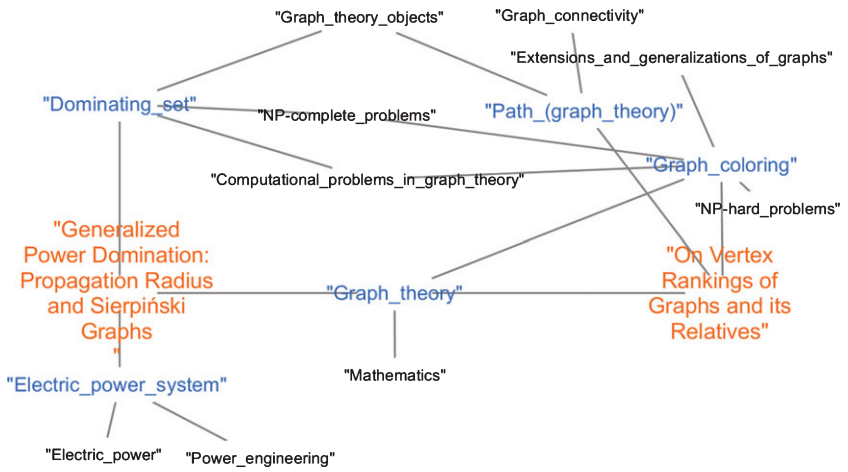

Fig. 2. An example of two papers and their relation via the http://purl.org/dc/terms/subject predicate. Large text (orange) is the paper title. The small text (blue) are the entities (i.e., E_i). Smaller text (black) are the entities properties (i.e., P_i). In this example, new knowledge is discovered by the link between Dominating_Set and Path_(graph_theory).

Table 1
Description of the uncleaned working dataset regarding the total number of entities, citations, and authors in the papers

	Entities	Total citations	Total authors
Min.	1	1	1
25%	4	1	2
50%	7	4	3
75%	11	15	4
Max.	60	9.6 K	2.9 K

Table 2
Statistics for the Entities and Authors frequency in the uncleaned dataset

	Mean	SD	Min.	25%	50%	75%	Max.
Entities	94	908	1	1	4	20	240 K
Authors	2	5	1	1	1	2	519

to remove outliers (e.g., papers with more than 2.9 K authors, 9.6 K citations). The cleaning is done in three steps.

First, since the dataset contains papers from year 1800, we compute the quartiles of the total of papers per year and filter in the 75% percentile (i.e.years with at least 1000 papers). The first year that satisfies this condition is 1973. Figure 3 presents a histogram of papers per year. Second, we remove papers with more than 20 authors and 100 citations. Third, we consider entities with a frequency above the median (i.e., entities that appear at least *four* times). After cleaning, the total number of papers is 385 K. Table 3 presents statistics of the clean dataset. Tables 4 and 5 present the top 10 authors, entities, and properties. Figure 4 shows popular co-occurrences for the top

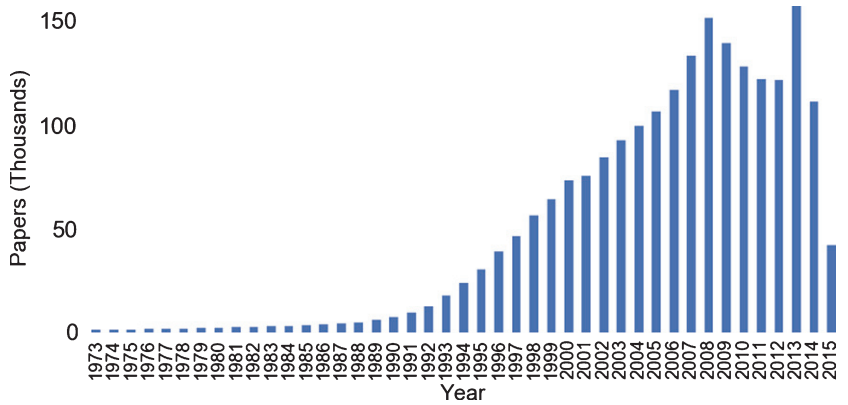

Fig. 3. Histogram of papers per year.

Table 3
Statistics of the cleaned dataset. The abstract length is in characters

	Authors	Citations	Abstract length	Entities
Mean	3	3	693	9
SD	2	3	198	5
Min.	1	1	0	1
25%	2	1	542	6
50%	3	3	667	9
75%	4	5	832	13
Max.	19	50	3.7K	53

Table 4
Most of the authors and entities are related to the fields of computer science, mathematics, and bio-informatics. Note that in this table the authors and entities are not related per row

Top 10 Authors	Top 10 Entities
Tania Vu	Algorithm
Jiawei Han	Protein
Nicholas G. Martin	Species
Gonzalo Navarro	Gene_expression
Lei Zhang	DNA
Hanspeter Seidel	Evolution
Terrence J. Sejnowski	Mathematical_optimization
Philip S. Yu	Graph_theory
Moshe Y. Vardi and	Mutation
Christos Faloutso	Gene

Table 5
DBpedia also links to other knowledge graphs like YAGO and other resources (e.g., purl – permanent URLs)

Top 10 Properties
http://purl.org/dc/terms/subject
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/2002/07/owl#Thing
http://dbpedia.org/class/yago/PhysicalEntity100001930
http://dbpedia.org/class/yago/Object100002684
http://dbpedia.org/class/yago/YagoLegalActorGeo
http://dbpedia.org/class/yago/YagoPermanentlyLocatedEntity
http://dbpedia.org/class/yago/Abstraction100002137
http://dbpedia.org/class/yago/Whole100003553
http://dbpedia.org/class/yago/YagoLegalActor

10 entities of Table 4 (i.e., entities mentioned in the same abstract).

3.3. Candidate sets

When researchers are creating a new paper, they can potentially cite any paper previously published. Theoretically, then, citation recommendations for a new paper should consider all past papers. To speed-up training, as well as the generation of recommendations, it is important to reduce the set of candidate papers to a relatively limited set of papers that include comparable proportions of true positives (i.e., cited papers) and true negatives (papers not cited).

We achieve this by considering papers similar to the one for which recommendations are produced. Specifically, we use Lucene to retrieve the K papers with abstracts that contain similar terms and entities according to standard tf-idf weighting scheme. We experiment with various values for K, and tune it to a moderate value that offers good recall levels. An alternative approach would include as candidates the cited papers of the K similar papers (as in [3]). However, we found empirically that this approach leads to large candidate sets, which render it impractical.

3.4. Training set

Training is set to build a model that answers the question: "given a list of pairs of papers i and candidates c_i, is c_i more likely to be cited by i than other candidates?". Textual and semantic features from abstracts, their candidate sets, and their citations are used to create a training set (Fig. 5). Specifically, each entry of the training set corresponds to a pair i, c_i of a paper i and its candidate citation c_i. The pairs are associated to item i so the model can learn the relevance of c_i in the list of candidates for i. The *predicted variable* is binary and encodes whether paper i cites paper c_i. The *predictor variables* capture the similarity of i and c_i regarding their textual and semantic features, as explained below.

3.4.1. Pairwise features

We build pairs between the papers I and the candidates $c_i \in CS$. We build the same features for the given abstract to predict.

MLT Score. The score returned by Lucene based on the similarity of the most relevant terms and entities.

Entities tf-idf. The sum of common entities in the candidate weighted by tf-idf using Equation 2.

Properties tf-idf. The sum of common entity properties in the candidate weighted by tf-idf using Equation 2.

Abstract tf-idf. The sum of common terms in the candidate weighted by tf-idf using Equation 2.

Combined tf-id. The sum of the entities tf-idf, properties tf-idf, and abstract tf-idf.

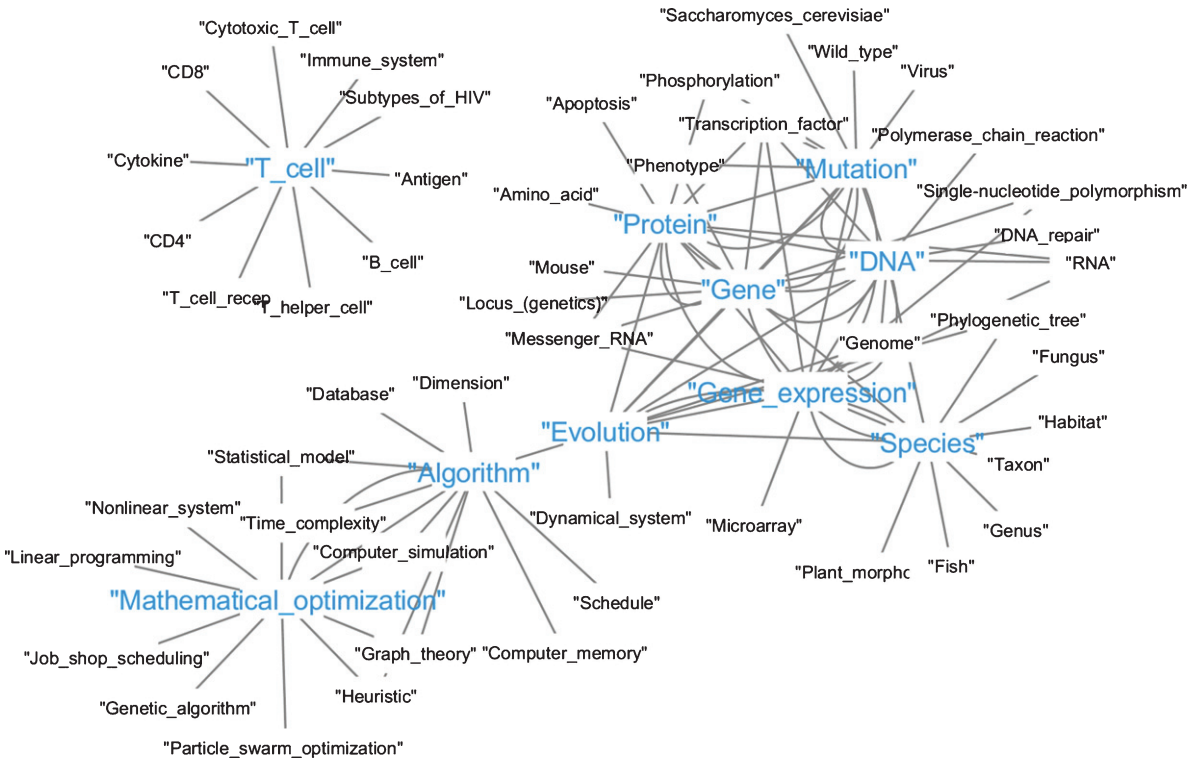

Fig. 4. Larger text (blue) are the top 10 entities. Smaller text (black) are entities co-occurrences (i.e., papers mentioning both entities). For simplicity, we show just 10 co-occurrences for each of the top entities.

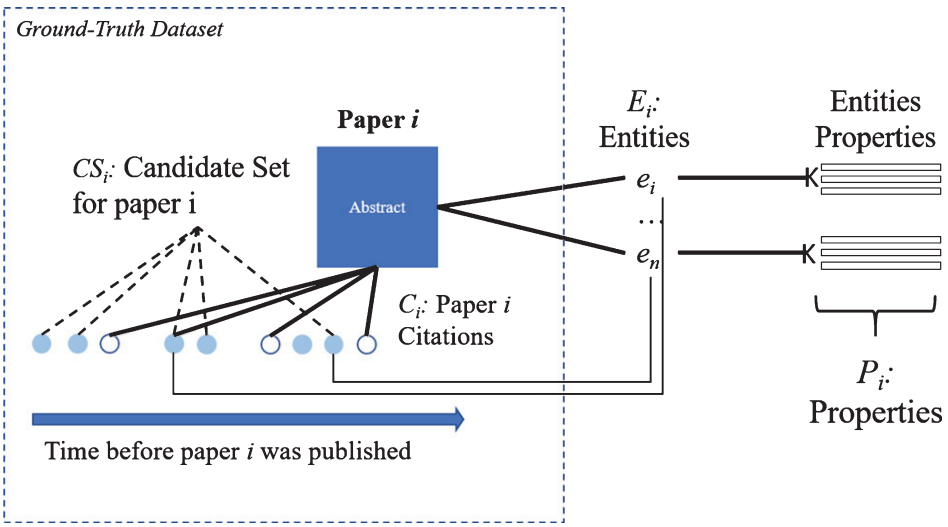

Fig. 5. Expanding the papers abstract with knowledge graphs creates a relation between the paper i and entities from DBpedia. These entities relate paper i to papers that share common entities. In our approach, we use these entities together with the abstract terms to generate a candidate paper CS that are likely to be cited by i. In a second step, we rank these candidate papers to build the Top-K recommendations.

Highest tf-idf. Categorical feature with the label of the highest tf-idf score (i.e., entities tf-idf, properties tf-idf, and abstract tf-idf).

Lowest idf-idf. Categorical feature with the label of the lowest tf-idf score (i.e., entities tf-idf, properties tf-idf, and abstract tf-idf).

Candidate Age. The years that have passed since the candidate paper was published. For example, if we are training in a paper from 2015 and the candidate is from 2005, the candidate's age is 10 years.

Indegree. The total number of papers citing the candidate as of the year of training or recommendation.

Indegree w/Decay. The indegree with a decay according to the candidate age

$$indegree_{decay}(c_i|a_i) = \frac{indegree_{c_i|a_i}}{\log(age_{c_i|a_i})}.$$

Equation 1 defines the relevance of c_i for paper i.

3.4.2. Learning to rank

Gradient boosting [34], boosted trees [35] and especially some variants of gradient boosted trees [36] are widely used methods for both classification and regression. LambdaMART is a gradient boosted tree based list-wise learning-to-rank algorithm which is a special version of RankNet. The model uses boosted trees optimized for nDCG instead of pairwise errors [31]. Since the occurring term and entity feature space is high dimensional and highly sparse the model needs high number of "weak" learners (shallow trees).

3.5. Prediction task

Given these notations and data, the problem of global citation recommendation is defined as follows. Let X be the training set of all triplets in our ground-truth dataset defined as

$$X = \bigcup_{i}^{I} \{< a_i, c, r(c_i|a_i) >\},$$

where

$$r(c_i|a_i) = \begin{cases} 1, & \text{if } c_i \exists C_i \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The goal is to learn a scoring function f that approximates the relevance $\hat{r}(c|a_i) = f(X)$ for $c \in CS_i$. The resulting $\hat{r}$ is used to sort the elements in CS_i and build Top K recommendations for a_i.

4. Evaluation methodology, metrics and baselines

4.1. Dataset split

We use two approaches for measuring the quality of the global citations recommendations produced by our approach. The first is *random split per year*. The motivation for this is to train a model using the papers published in a year, hold part of the papers as testing data and then evaluate the recommendations on papers written in the same year. For this purpose, we split the data into three parts. 80% of the papers are used as training set, 20% as testing set, and from the testing set, we take 10% as a validation set. The second approach is *evaluating on the next year*. The motivation for this is that we want to know how good is the approach for recommending on the next year. Table 6 presents the numbers of papers per year, number of papers, and pairs used for training,

Table 6
For building the pairs, we consider papers with at least 3 citations. Then, we split the total number of papers in three datasets: training, testing and validation. For each of these papers we build pairs of abstract-candidate and compute their features

Year	Total papers	Training papers	Training pairs	Test papers	Test pairs	Validation papers	Validation pairs
1996	202	162	55.2 K	36	12.4 K	4	1.3 K
1997	406	325	117.5 K	73	26.4 K	8	2.8 K
1998	758	607	225.4 K	136	50.0 K	15	6.4 K
1999	1.2 K	967	372.7 K	217	86.5 K	24	9.7 K
2000	1.7 K	1.4 K	561.4 K	310	128.9 K	34	14.0 K
2001	2.3 K	1.8 K	801.8 K	414	178.4 K	45	19.4 K
2002	3.0 K	2.4 K	1.1 M	539	244.5 K	59	27.3 K
2003	4.1 K	3.3 K	1.6 M	747	349.2 K	82	38.8 K
2004	5.0 K	4.0 K	1.9 M	900	435.7 K	99	49.1 K
2005	6.2 K	5.0 K	2.5 M	1.1 K	554.6 K	124	62.5 K
2006	7.4 K	5.9 K	3.0 M	1.3 K	673.2 K	147	75.9 K
2007	8.4 K	6.7 K	3.4 M	1.5 K	769.2 K	168	86.4 K
2008	10.4 K	8.3 K	4.3 M	1.9 K	959.3 K	207	107.6 K
2009	11.2 K	9.0 K	4.7 M	2.0 K	1.1 M	223	116.2 K
2010	10.5 K	8.4 K	4.4 M	1.9 K	982.3 K	209	109.8 K
2011	10.9 K	8.7 K	4.6 M	2.0 K	1.0 M	217	113.0 K
2012	9.5 K	7.6 K	4.1 M	1.7 K	916.4 K	189	100.1 K

Table 7
The *LambdaMART* models follow our proposed approach. For the *ATF* and *MLT* baselines we sort the papers in the candidate set according to this metric without using LambdaMART

Model	Acronym	Features
Lucene's More Like This	MLT	Relevant terms in abstract ∪ entities
Abstract tf-idf Score	ATF	Abstract tf-idf score as Equation 2
LambaMART No Entities/Properties	LM-w/o	Candidate age, indegree w/ decay, indegree, abstract tf-idf
LambdaMART All	LM-all	All the features described in the Subsection 3.4.1

Table 8
Features used for building the candidate sets

Features	Recall@10	Recall@20	Recall@50	Recall@100	Recall@500	Recall@1000
Abstract	0.37	0.41	0.47	0.52	0.62	0.66
Entities	0.37	0.40	0.46	0.50	0.59	0.62
Abstract ∪ Entities	**0.39**	**0.43**	**0.49**	**0.54**	**0.64**	**0.67**

testing, and validating. For the *evaluating on the next year*, we take all of its pairs to build and evaluate the recommendations.

4.2. Metric

As performance metric we use nDCG@10 defined as

$$nDCG@10 = \frac{\sum_{i}^{10} \frac{2^{rel_i}-1}{log_2(i+1)}}{IDCG},$$

where rel_i is 1 if the recommended paper is cited and $IDCG$ is the ideal ordering of the ranking.

4.3. Baselines

4.3.1. Weighted tf-idf score

For this baseline, we compute the weighted tf-idf of the terms in the candidate's abstract. Then, we sum the tf-idf of the common terms between the candidate and the given abstract. The weighted tf-idf [37] is defined as

$$w_{t,d} = (1 + \log f_{t,d}) \times \log_{10}\left(\frac{N}{df_t}\right)$$

where N is the number of papers, $f_{t,d}$ is the frequency of the term in the document and df_t is the number of documents containing t. The tf-idf score for a candidate given an abstract is

$$score(i, c) = \sum_{t \in i \cup c} tf.idf_{t,c}. \quad (2)$$

4.3.2. Lucene's more like this

This baseline is the proposed approach for candidate sampling. That is, we rank the candidates according to the *Lucene's More Like This* score on terms in the abstract and entities.

5. Experimental results

5.1. Experimental setting

We used Microsoft's LightGBM[9] implementation of LambdaMART. In the validation set, we fine-tune the parameters using Bayesian Optimization [38]. We found the following parameters to perform consistently well: 1000 trees, 30 leaves, and high minimum number of objects in a leaf (100). For Lucene's More Like we used grid search to fine-tune its parameters. We used *ElasticSearch*[10] as an interface to Lucene. The code for generating the candidate sets, pairwise features, model training, and predictions was executed on a single machine with 224GB of RAM, 48 cores, and run for a day.

Table 7 presents the different variants of the baselines and LambdaMART used in our experiments.

5.2. Empirical results

Table 8 presents the coverage of the proposed candidate set sampling method at different set sizes.

Table 9 presents the results for the evaluation on *random split per year*. And, Table 10 the results for evaluation *on the next year*.

The feature importance for the models is presented in Fig. 6.

[9]https://github.com/Microsoft/LightGBM/

[10]https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html

Table 9
nDCG@10 for the models evaluated on the year random split. The numbers in parenthesis indicate the gain of the *LM-all* in comparison to the baseline

Split Year	LM-all	MLT	ATF	LM-w/o
1996	0.402	0.344 (+16.9%)	0.262 (+53.7%)	0.292 (+37.7%)
1997	0.449	0.409 (+10.0%)	0.326 (+38.1%)	0.307 (+46.5%)
1998	0.390	0.390 (+0.0%)	0.300 (+29.8%)	0.291 (+34.0%)
1999	0.401	0.364 (+10.2%)	0.274 (+46.5%)	0.292 (+37.2%)
2000	0.411	0.391 (+5.0%)	0.261 (+57.3%)	0.282 (+45.8%)
2001	0.382	0.347 (+10.2%)	0.268 (+42.6%)	0.289 (+32.0%)
2002	0.368	0.330 (+11.5%)	0.252 (+46.1%)	0.284 (+29.5%)
2003	0.380	0.347 (+9.5%)	0.245 (+54.7%)	0.274 (+38.4%)
2004	0.335	0.308 (+8.8%)	0.211 (+58.6%)	0.236 (+42.2%)
2005	0.327	0.297 (+10.1%)	0.212 (+54.0%)	0.250 (+30.5%)
2006	0.318	0.293 (+8.6%)	0.207 (+53.6%)	0.235 (+35.2%)
2007	0.316	0.290 (+8.9%)	0.194 (+62.5%)	0.226 (+40.1%)
2008	0.310	0.286 (+8.6%)	0.194 (+60.1%)	0.223 (+39.1%)
2009	0.300	0.265 (+13.2%)	0.189 (+58.6%)	0.228 (+31.8%)
2010	0.316	0.286 (+10.6%)	0.194 (+62.9%)	0.229 (+38.3%)
2011	0.306	0.281 (+8.9%)	0.194 (+57.9%)	0.218 (+40.0%)
2012	0.303	0.268 (+12.9%)	0.174 (+74.2%)	0.211 (+43.4%)
2013	0.292	0.252 (+15.8%)	0.167 (+74.4%)	0.207 (+41.0%)
2014	0.294	0.251 (+17.2%)	0.175 (+68.0%)	0.215 (+36.8%)
2015	0.256	0.212 (+21.0%)	0.151 (+69.7%)	0.193 (+33.2%)

Table 10
nDCG@10 for the models evaluated on the following year of the training. The numbers in parenthesis indicate the gain of the *LM-all* in comparison to the baseline

Eval. Year	LM-all	MLT	ATF	LM-w/o
1997	0.458	0.373 (+22.7%)	0.287 (+59.7%)	0.338 (+35.7%)
1998	0.420	0.369 (+13.7%)	0.260 (+61.2%)	0.308 (+36.4%)
1999	0.427	0.374 (+13.9%)	0.280 (+52.2%)	0.316 (+34.9%)
2000	0.417	0.384 (+8.6%)	0.271 (+53.8%)	0.314 (+33.0%)
2001	0.391	0.356 (+10.0%)	0.263 (+48.6%)	0.295 (+32.8%)
2002	0.378	0.336 (+12.6%)	0.253 (+49.7%)	0.286 (+32.3%)
2003	0.365	0.327 (+11.5%)	0.238 (+53.4%)	0.271 (+34.6%)
2004	0.346	0.312 (+10.8%)	0.222 (+56.1%)	0.257 (+34.7%)
2005	0.334	0.301 (+11.1%)	0.218 (+53.7%)	0.252 (+32.5%)
2006	0.325	0.292 (+11.6%)	0.209 (+55.6%)	0.241 (+34.7%)
2007	0.324	0.290 (+11.7%)	0.202 (+60.4%)	0.238 (+36.1%)
2008	0.313	0.279 (+12.3%)	0.196 (+59.7%)	0.230 (+36.1%)
2009	0.312	0.276 (+13.1%)	0.192 (+62.4%)	0.228 (+37.2%)
2010	0.318	0.284 (+12.0%)	0.198 (+60.5%)	0.232 (+37.5%)
2011	0.306	0.271 (+12.7%)	0.188 (+62.3%)	0.221 (+38.2%)
2012	0.300	0.263 (+14.0%)	0.177 (+69.4%)	0.215 (+39.8%)
2013	0.296	0.255 (+16.2%)	0.176 (+68.5%)	0.214 (+38.1%)
2014	0.295	0.259 (+14.0%)	0.176 (+67.7%)	0.215 (+37.1%)
2015	0.281	0.235 (+19.8%)	0.168 (+67.2%)	0.212 (+32.6%)

5.3. Findings

The following findings summarize the main outcomes of our research work based on the presented experimental results.

Finding 1: *Expanding Semantic Features with DBpedia*. DBpedia Spotlight allowed us to expand the semantic features of the text. From the annotated DBpedia entities we extracted two new features: *entities* and *entities properties*. These features contributed to building Top K global citation recommendations as indicated in the next findings.

Finding 2: *Candidate Set*. Table 8 shows that *Lucene's More Like This* method has a higher recall when the relevant terms in the abstract and the entities are combined. This means that the entities help Lucene to formulate a better query to retrieve possible relevant candidates.

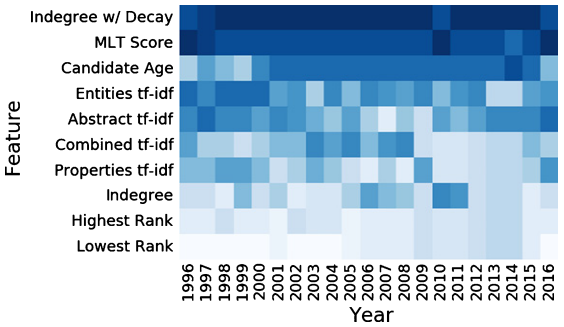

Fig. 6. The feature importance per year. The features are sorted by the times they appear in the LambdaMART model. The features are sorted from top to bottom, and stronger color means a greater contribution to the model.

Finding 3: *Building Top-K Recommendations*. Table 9 and 10 show that our proposed approach performs well in papers published during the same year of training and also in the next year. These results confirm that the approach of building the candidate set, pairwise abstract-candidate features, and fitting LambdaMART to learn the relevance, leads to a higher nDCG@10 than the baselines.

Finding 4: *Feature Importance*. Tables 9 and 10 show that the model *LM-all* performs significantly better than *LM-w/o*. Moreover, Fig. 6 shows that the *Entities tf-idf* is of high importance, as it is among the most frequently chosen by the LambdaMART model. These results show that the expanded semantic features are helping the model to differentiate what makes a candidate to be relevant for the given abstract.

6. Conclusions

When modern researchers are searching for the literature of a new paper, they could save time by receiving a list of recommended papers.

Our research work presents empirical findings on using knowledge graphs to build global citation recommendations. To the best of our knowledge, this is the first study that involves semantic features of knowledge graphs in building the candidate set and training the ranking model for this task. Our experiments show that these semantic features contribute significantly and help outperform text-based baseline.

Given the importance of semantic features, an obvious direction of future work is to incorporate them in recommender systems tailored to different settings (e.g., for local recommendations). Moreover, as seen in the Section 3.2, most of the papers are related to areas of computer science, mathematics, and natural sciences. Therefore, another direction of future work would be to build 'vertical' recommender systems, that are fit to the semantics of different fields (e.g., dblp, PubMed). Finally, trying different weighting methods (e.g., Okapi BM25) could help diminish the effect of incorrectly annotated entities.

Acknowledgments

Frederick Ayala-Gómez was supported by the *Mexican National Council for Science and Technology* (Consejo Nacional de Ciencia y Tecnología–CONACYT), and by the European Institute of Innovation and Technology (EIT) Digital Doctoral School. This publication was also supported from the *EU H2020* research and innovation program under grant agreement *Streamline No 688191* and the "Big Data—Momentum" grant of the Hungarian Academy of Sciences. Aristides Gionis is supported by the Academy of Finland project "Nestor" (286211) and the EC H2020 RIA project "SoBigData" (654024).

References

[1] M. Mabe, The growth and number of journals, *Serials* **16**(2) (2003), 191–197.

[2] P.O. Larsen and I.M. Von, The rate of growth in scientific publication and the decline in coverage provided by science citation index, *Scientometrics* **84**(3) (2010), 575–603.

[3] Q. He, J. Pei, D. Kifer, P. Mitra and L. Giles, Context-aware citation recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW'10*, New York, NY, USA, 2010, pp. 421–430. ACM.

[4] F. Ricci, L. Rokach and B. Shapira, *Recommender Systems Handbook*. Springer Publishing Company, Incorporated, 2nd edition, 2015.

[5] J. Beel, B. Gipp, S. Langer and C. Breitinger, Research-paper recommender systems: a literature survey, *International Journal on Digital Libraries* **17**(4) (2016), 305–338.

[6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, Dbpedia: A nucleus for a web of open data. The semantic web, 2007, pp. 722–735.

[7] P.N. Mendes, M. Jakob, A. García-Silva and C. Bizer, Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, 2011, pp. 1–8. ACM.

[8] L.H. Peter, A statistical approach to mechanized encoding and searching of literary information, *IBM Journal of Research and Development* **1**(4) (1957), 309–317.

[9] W. Huang, Z. Wu, P. Mitra and C.L. Giles, Refseer: A citation recommendation system. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2014, pp. 371–374. IEEE Press.

[10] S. Kataria, P. Mitra and S. Bhatia, Utilizing context in generative bayesian models for linked corpus, In *AAAI*, volume 10, 2010, p. 1.

[11] H. Li, I. Councill, W.-C. Lee and C.L. Giles, Citeseerx: an architecture and web service design for an academic document search engine. In *Proceedings of the 15th International Conference on World Wide Web*, 2006, pp. 883–884. ACM.

[12] M. Hagen, A. Beyer, T. Gollub, K. Komlossy and B. Stein, Supporting scholarly search with keyqueries. In *European Conference on Information Retrieval*, Springer, 2016, pp. 507–520.

[13] S.E. Middleton, D.C. De Roure and N.R. Shadbolt, Capturing knowledge of user preferences: Ontologies in recommender systems. In *Proceedings of the 1st International Conference on Knowledge Capture, K-CAP '01*, New York, NY, USA, 2001, pp. 100–107. ACM.

[14] I.C. Paraschiv, M. Dascalu, P. Dessus, S. Trausan-Matu and D.S. McNamara, A paper recommendation system with readerbench: the graphical visualization of semantically related papers and concepts. In *State-of the-Art and Future Directions of Smart Learning*, Springer, 2016, pp. 445–451.

[15] J. Beel, S. Langer, M. Genzmehr and A. Nürnberger, Introducing docear's research paper recommender system. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, ACM, 2013, pp. 459–460.

[16] M. Baez, D. Mirylenka and C. Parra, Understanding and supporting search for scholarly knowledge. *Proceeding of the 7th European Computer Science Summit*, 2011, pp. 1–8.

[17] H. Claudia, Trust-based recommendations for publications: A multi-layer network approach, *TCDL Bulletin* **2**(2) (2006), 190–201.

[18] Y. Liang, Q. Li and T. Qian, Finding relevant papers based on citation relations. In *International Conference on Web-Age Information Management*, Springer, 2011, pp. 403–414.

[19] A. Arnold and W.W. Cohen, Information extraction as link prediction: Using curated citation networks to improve gene detection. In *International Conference on Wireless Algorithms, Systems, and Applications*, Springer, 2009, pp. 541–550.

[20] K. Jack, Mahout becomes a researcher: large scale recommendations at mendeley. In *Presentation at Big Data Week Conferences*, 2012.

[21] D. Zhou, S. Zhu, K. Yu, X. Song, B.L. Tseng, H. Zha and C.L. Giles, Learning multiple graphs for document recommendations. In *Proceedings of the 17th International Conference on World Wide Web*, ACM, 2008, pp. 141–150.

[22] T. Strohman, W.B. Croft and D. Jensen, Recommending citations for academic papers. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2007, pp. 705–706.

[23] S. Bethard and D. Jurafsky, Who should i cite: learning literature search models from citation behavior. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ACM, 2010, pp. 609–618.

[24] J. Wu, C. Liang, H. Yang and C.L. Giles, Citeseerx data: Semanticizing scholarly papers. In *Proceedings of the International Workshop on Semantic Big Data, SBD '16*, New York, NY, USA, 2016, pp. 2:1–2:6. ACM.

[25] X. Cheng and R. Dan, Relational inference for wikification, *Urbana* **51**(61801) (2013), 16–58.

[26] L. Ratinov, D. Roth, D. Downey and M. Anderson, Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11*, Stroudsburg, PA, USA, Association for Computational Linguistics, 2011, pp. 1375–1384.

[27] T.D. Noia, V.C. Ostuni, P. Tomeo and E.D. Sciascio, Sprank: Semantic path-based ranking for top-n recommendations using linked open data, *ACM Transactions on Intelligent Systems and Technology (TIST)* **8**(1) (2016), 9.

[28] Y. Koren, R. Bell, C. Volinsky, et al., Matrix factorization techniques for recommender systems, *Computer* **42**(8) (2009), 30–37.

[29] X. Ning and G. Karypis, Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, 2011, pp. 497–506. IEEE.

[30] A. Alzoghbi, V.A.A. Ayala, P.M. Fischer and G. Lausen, Learning-to-rank in research paper cbf recommendation: Leveraging irrelevant papers. In *CBRec-Sys@ RecSys*, 2016, pp. 43–46.

[31] C.J.C. Burges, From ranknet to lambdarank to lambdamart: An overview, *Learning* **11**(23-581) (2010), 81.

[32] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J.P. Hsu and K. Wang, An overview of Microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 243–246. ACM.

[33] A. Cohen, FuzzyWuzzy: Fuzzy string matching in python, 2017. [Online; accessed July 2017].

[34] J.H. Friedman, Greedy function approximation: a gradient boosting machine, *Annals of Statistics* (2001), 1189–1232.

[35] Y. Freund and R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, Springer, 1995, pp. 23–37.

[36] T. Chen and T. He, Xgboost: extreme gradient boosting, R package version 0.4-2, 2015.

[37] D. Christopher, Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[38] R. Martinez-Cantin, Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits, *J Mach Learn Res* **15**(1) (2014), 3735–3739.