Information Sciences 221 (2013) 245–261

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Using semi-structured data for assessing research paper similarity

Germán Hurtado Martín [a,b,*], Steven Schockaert [c], Chris Cornelis [b,d], Helga Naessens [a]

[a] Dept. of Industrial Engineering, University College Ghent, Belgium
[b] Dept. of Applied Mathematics and Computer Science, Ghent University, Belgium
[c] School of Computer Science & Informatics, Cardiff University, UK
[d] Dept. of Computer Science and Artificial Intelligence, University of Granada, Spain

ARTICLE INFO

Article history:
Received 8 December 2011
Received in revised form 24 May 2012
Accepted 26 September 2012
Available online 6 October 2012

Keywords:
Document similarity
Semi-structured document
Language modeling
Latent Dirichlet Allocation

ABSTRACT

The task of assessing the similarity of research papers is of interest in a variety of application contexts. It is a challenging task, however, as the full text of the papers is often not available, and similarity needs to be determined based on the papers' abstract, and some additional features such as their authors, keywords, and the journals in which they were published. Our work explores several methods to exploit this information, first by using methods based on the vector space model and then by adapting language modeling techniques to this end. In the first case, in addition to a number of standard approaches we experiment with the use of a form of explicit semantic analysis. In the second case, the basic strategy we pursue is to augment the information contained in the abstract by interpolating the corresponding language model with language models for the authors, keywords and journal of the paper. This strategy is then extended by revealing the latent topic structure of the collection using an adaptation of Latent Dirichlet Allocation, in which the keywords that were provided by the authors are used to guide the process. Experimental analysis shows that a well-considered use of these techniques significantly improves the results of the standard vector space model approach.

1. Introduction

Due to the rapidly growing number of published research results, searching for relevant papers can become a tedious task for researchers. In order to mitigate this problem, several solutions have been proposed, such as scientific article recommender systems [4,24,29,8] or dedicated search engines such as Google Scholar. At the core of such systems lies the ability to measure to what extent two papers are similar, e.g. to find out whether a paper is similar to papers that are known to be of interest to the user, to explicitly allow users to find "Related articles" (as in Google Scholar), or to ensure that the list of search results that is presented to the user is sufficiently novel and diverse [6]. To find out whether two articles are similar, content-based approaches can be complemented with collaborative filtering techniques (e.g. based on CiteULike.org or Bibsonomy.org) or citation analysis (e.g. PageRank, HITS, etc.). While the latter are well-studied, content-based approaches are usually limited to baseline techniques such as using the cosine similarity between vector representations of the abstracts.

Comparing research papers is complicated by the fact that their full text is often not publicly available, and only semi-structured document information containing the abstract along with some document features such as keywords, authors, or journal can be accessed. The challenge thus becomes to make optimal use of this limited amount of information. A first option is to use these features in the vector space model, either separately or combined in some way. Alternatively, language

* Corresponding author at: Dept. of Industrial Engineering, University College Ghent, Belgium.
E-mail address: german.hurtadomartin@ugent.be (G. Hurtado Martín).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.09.044

modeling techniques can be considered, as they have already been shown to perform well for comparing short text snippets [14,30].

Our main goal in this paper is to study to what extent such additional, semi-structured information can be used to compare research paper abstracts, and how we can make use of it in the context of either the vector space model or a language modeling approach. For the vector space model, we first consider the traditional tf–idf approach as a baseline method, and then investigate the potential of the idea of explicit semantic analysis. In particular, we adapt a method from [10], representing each document as a vector of keywords, a vector of authors, or a vector of journals. By abstracting away from the individual terms that appear in a document, and rather describing it in terms of how strongly it is related to e.g. a given keyword, we can hope to overcome problems of vocabulary mismatch that hamper the baseline method. For language modeling, on the one hand, we consider the idea of estimating language models for document features such as keywords, authors, and journal, and estimate a language model for the overall article by interpolating these models, an idea which has already proven useful for expert finding [38]. Furthermore, we use Latent Dirichlet Allocation (LDA) to discover latent topics in the documents, and further improve the language model of an article based on the revealed topical information. To improve the performance of the standard LDA method, we replace its random initialization by an initialization which is based on the keywords that have been assigned to each paper. The main underlying idea is that a topic can be identified with a cluster of keywords.

The remainder of this paper is structured as follows. After reviewing the related work in Section 2, we study two methods based on the vector space model to measure article similarity in Section 3, while in Section 4 we propose a number of methods based on language modeling. In Section 5 we explain the details concerning our experimental set-up, and present and discuss the obtained results. The main conclusions are summarized in Section 6.

Note that this paper extends and improves on the approaches that were presented in [15] and [16]. In particular, it improves on both previous works in the way the various methods are evaluated. We have substantially expanded the analysis of the results, and use a more representative test set than before. In addition, a number of key improvements to the language modeling methods have been added, including the use of communities and the combination of features in LDA methods (resulting in the *LMe* methods that offer the best performance, as shown by the results).

2. Related work

Language models are a relatively recent approach to information retrieval, and are typically seen as an alternative to the traditional methods based on the vector space model. The language modeling approach is based on the assumption that a document has been generated using some kind of probabilistic model. To estimate the relevance of a query to a document, we then try to estimate the underlying probabilistic model, primarily based on the terms that occur in it, and then compare the query to that model, rather than to the actual document. Most current work builds on the initial approach by Ponte and Croft [28]. The most common way to improve language models is to improve the smoothing method. The basic idea of smoothing is to estimate the probability that a term is generated by the language model underlying a document not only from the terms that occur in the document itself, but also from the terms that occur in the rest of the collection. It is used to lessen the impact of common words (not unlike the idea of inverse document frequency in the vector space model), and to ensure that only non-zero probabilities are used. A comprehensive overview of the most common smoothing methods can be found in [36]. A number of authors have investigated smoothing methods that go beyond the standard approaches. For instance, [22] combines Dirichlet smoothing with bigrams, instead of the unigrams typically used, and the collection used for smoothing is expanded with external corpora, for the task of spontaneous speech retrieval. Deng et al. [7] follow a somehow inverse approach and apply smoothing based only on subsets of the collection corresponding to a specific community of experts. Different smoothing strategies are found in the literature precisely for this task of expert finding. Karimzadehgan et al. [18] and Petkova and Croft [26] try to improve smoothing by interpolating models, expanding the idea originally proposed by [17] on which we have partially built our approach. The idea of interpolating different language models was used in a particularly comprehensive way in [26]: to represent an expert, a model is estimated for his mails, another model for his papers, etc., and then they are interpolated; at the same time, in order to model the mails, a model can be created for the body of the mails, another model for the subject headers, etc. Mimno and McCallum [25] evaluate, for the same task, models that combine author-based information with Dirichlet smoothing. Finally, [37] also proposes the interpolation of several models to discover new expertise.

It is interesting to see that, as in our case, efforts to improve language modeling often lead to the use of Latent Dirichlet Allocation [2]. Examples of this are the already mentioned methods of [25,37]. As we will discuss in Section 4.2.1, the topics underlying a particular collection of documents (and a document itself) can be discovered by using LDA. These topic models have gained a lot of popularity in the last years and have been used in a vast diversity of tasks such as tag recommendation [20], measuring the influence of Twitter users [35], or text classification [27]. The basic form of LDA does not suffice in many cases, however. While for some problems it is enough to adapt the distributions used by the algorithm [1], most of the solutions involve changes in the way the estimated probabilities are computed and, depending on the task, different kinds of extra information are incorporated. For example, the chronological order of the documents can be taken into account to discover topic trends [5]; on the other hand [34] considers the intuition of authors usually writing about the same topics, and add information about authors to create author-topic models, which in turn have been improved as well [12,25]. A different

approach consists in improving language models by using document labels, such as scores or tags, which can be used as a kind of supervision [3], or be associated with the topics in direct correspondence [31]. The approach of Kataria et al. [19] could also be included in this group, as they use entities, annotations, and classifications from Wikipedia to construct better models. One of the methods proposed in this latter work has some similarities with ours, as the number of times a word is assigned to a Wikipedia topic is used in LDA in a manner comparable to our *LM0e* method (Section 4.3). However, our strategy uses no external sources of information, but only what is already in the document. Finally, it should be noted that the LDA topic models cannot only be improved by feeding them with additional information, but also by improving the initialization of the Gibbs sampling method that is typically used. This idea appears to have received little attention in the literature, and will be explored in methods LM2 and LM2e (Section 4.4).

3. Vector space model

In this section we discuss two methods based on the vector space model to assess paper similarity using the information commonly available for a research paper: abstract, keywords, authors, and journal. First we propose an approach which makes use of tf–idf (term frequency–inverse document frequency), and then another one based on an indirect indexing scheme known as explicit semantic analysis [10].

3.1. Baseline

A simple way to measure the similarity of two papers is by comparing their abstracts in the vector space model: each paper is represented as a vector, with each component corresponding to a term (unigram) occurring in the collection. To convert a document into a vector, the stopwords[1] are first removed; we do not use stemming. Then, to calculate the weight for each term t_i in the abstract of document d, the tf–idf scoring technique [32] is used:

$$tfidf(t_i, d) = \frac{n(t_i, d)}{|d|} \cdot log \frac{|\mathcal{C}|}{|\{d_j : t_i \in d_j\}| + 1} \tag{1}$$

where $n(t_i, d)$ is the number of occurrences of t_i in the abstract of d, $|d|$ is the total number of terms in the abstract of d, $|\mathcal{C}|$ is the number of abstracts in the collection, and $|\{d_j: t_i \in d_j\}|$ is the number of abstracts in the collection that contain t_i. Two vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ corresponding to different papers can then be compared using standard similarity measures such as the cosine, generalized Jaccard, extended Jaccard, and Dice similarity, defined respectively by

$$sim_c(\mathbf{d_1}, \mathbf{d_2}) = \frac{\mathbf{d_1} \cdot \mathbf{d_2}}{\|d_1\| \cdot \|d_2\|} \tag{2}$$

$$sim_{gj}(\mathbf{d_1}, \mathbf{d_2}) = \frac{\sum_k min(d_{1_k}, d_{2_k})}{\sum_k max(d_{1_k}, d_{2_k})} \tag{3}$$

$$sim_{ej}(\mathbf{d_1}, \mathbf{d_2}) = \frac{\sum_k min(d_{1_k}, d_{2_k})}{\|d_1\|^2 + \|d_2\|^2 - (\mathbf{d_1} \cdot \mathbf{d_2})} \tag{4}$$

$$sim_d(\mathbf{d_1}, \mathbf{d_2}) = \frac{2(\mathbf{d_1} \cdot \mathbf{d_2})}{\|d_1\|^2 \cdot \|d_2\|^2} \tag{5}$$

where $\mathbf{d_1} \cdot \mathbf{d_2}$ denotes the scalar product and $\|.\|$ the Euclidean norm. We refer to the method that combines tf–idf on the abstract with these four similarity measures as *abstract*.

We also consider vector representations that are based on the keywords that have been assigned to the documents, thus ignoring the actual terms of the abstract (method *keywords*). Each component then represents a keyword from the collection. However, since each keyword occurs only once in a document, the tf–idf formula used in this case degenerates to:

$$tfidf(t_i, d) = \frac{1}{|d|} \cdot log \frac{|\mathcal{C}|}{|\{d_j : t_i \in d_j\}| + 1} \tag{6}$$

where $|d|$ is now the number of keywords assigned to the document, instead of the number of terms in the abstract. Unlike in the method *abstract*, where the terms are unigrams (individual terms), here we consider the whole keywords, which may be multigrams (e.g. "recommender system").

3.2. Explicit semantic analysis

A problem with the previous methods is that only one feature (keywords or abstract) is used at a time. Valuable information is thus ignored, especially in the *keywords* method which does not use the abstracts. In order to use the keywords without ignoring the information from the abstract, we propose an alternative scheme which we refer to as explicit semantic

[1] The list of stopwords we have used for the experiments was taken from http://snowball.tartarus.org/algorithms/english/stop.txt, expanded with the following extra terms: *almost*, *either*, *without*, and *neither*.

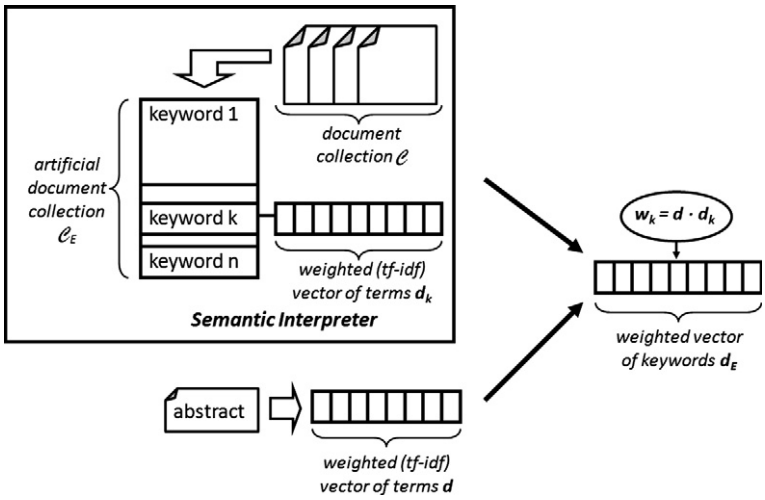

Fig. 1. Generation of the ESA vector $\mathbf{d_E}$ of a document.

analysis (ESA), as it is analogous to the approach from [10]. In this scheme, a new vector representation $\mathbf{d_E}$ is defined for each document d, where $\mathbf{d_E}$ has one component for every keyword k in the collection. The idea is that each component of the vector should reflect how related the document is with the concept represented by the corresponding keyword.

Let $\mathbf{d}$ be the vector obtained from method *abstract*. In addition, we consider a vector to represent each keyword (and, therefore, each concept). In order to build such a vector, a new collection $\mathcal{C}_E$ of artificial documents is considered, with one document d_k for each keyword, which consists of the concatenation of the abstracts of the documents from the original collection to which keyword k was assigned. Then, a weighted vector $\mathbf{d_k}$ is considered for each d_k, with each component corresponding to a term in $\mathcal{C}_E$ and where the weights are the tf–idf scores calculated w.r.t. $\mathcal{C}_E$. Thus, $\mathbf{d_k}$ represents the concept corresponding to keyword k in the same way that $\mathbf{d}$ represents document d. Finally, after normalizing both $\mathbf{d}$ and $\mathbf{d_k}$, they are compared to calculate the weight w_k in $\mathbf{d_E}$ of the component corresponding to keyword k:

$$w_k = \mathbf{d} \cdot \mathbf{d_k} \tag{7}$$

Fig. 1 summarizes this process. For a detailed example we refer to Appendix A. The $\mathbf{d_E}$ vectors can be compared by using any of the similarity measures defined in Section 3.1.

We further refer to this method as *ESA-kws*. Similar methods are considered in which vector components refer to authors (*ESA-aut*) or to journals (*ESA-jou*), where, instead of $\mathbf{d_k}$, a weighted vector $\mathbf{d_a}$ (for authors) or $\mathbf{d_j}$ (for journals) is used. In these cases, the collection $\mathcal{C}_E$ of artificial documents is built by considering a document d_a for each author (resp. d_j for each journal), which consists of the concatenation of the abstracts of the documents from the original collection to which author a (resp. journal j) is associated. For efficiency and robustness, only authors are considered that appear in at least four papers of the collection in the *ESA-aut* method, and only keywords that appear in at least six papers in the *ESA-kw* method.

4. Language modeling

4.1. Baseline

As an alternative to the approaches based on the vector space model, we consider estimating unigram language models [28] for each document, and calculating their divergence. A document d is then assumed to be generated by a given probabilistic model D. This model is estimated from the terms that occur in the abstract of d (and the rest of the abstracts in the collection). Using Jelinek-Mercer smoothing [36], the probability that model D generates term w is given by:

$$D(w) = \lambda P(w|d) + (1-\lambda)P(w|\mathcal{C}) \tag{8}$$

where $\mathcal{C}$ is again the whole collection of abstracts, and λ controls the weight given to the smoothing term $P(w|\mathcal{C})$. The probabilities $P(w|d)$ and $P(w|\mathcal{C})$ are estimated using maximum likelihood, i.e. $P(w|d)$ is the fraction of occurrences of term w in the abstract of document d, and $P(w|\mathcal{C})$ is the fraction of occurrences of term w in the collection. Once the models D_1 and D_2 corresponding to two documents d_1 and d_2 are estimated, we measure their difference using the Kullback–Leibler divergence [21], defined by

$$KLD(D_1||D_2) = \sum_w D_1(w) log \frac{D_1(w)}{D_2(w)} \tag{9}$$

Note that $KLD(D_1||D_2)$ is not equal to $KLD(D_2||D_1)$ in general. If a symmetric measure is desired, a well-known and popular alternative is Jensen–Shannon divergence [9], where the models are first compared to an average model $(D_1 + D_2)/2$ and then the mean of both divergences is calculated:

$$JSD(D_1||D_2) = \frac{KLD(D_1||\frac{D_1+D_2}{2})}{2} + \frac{KLD(D_2||\frac{D_1+D_2}{2})}{2} \tag{10}$$

In the remainder of this section we consider different ideas to improve this basic language modeling approach.

4.2. Language model interpolation

The probabilities in the model of a document are calculated using the abstracts in the collection. However, given the short length of the abstracts, it is important to make maximal use of all the available information, i.e. to also consider the keywords k, authors a, and journal j of the paper. In particular, the idea of interpolating language models, which underlies Jelinek-Mercer smoothing, can be generalized:

$$D(w) = \lambda_1 P(w|d) + \lambda_2 P(w|k) + \lambda_3 P(w|a) + \lambda_4 P(w|j) + \lambda_5 P(w|\mathcal{C}) \tag{11}$$

with $\sum_i \lambda_i = 1$. Interpolation of language models has also been used for example in [38] for the task of expert finding, integrating several aspects of a document in a model. In order to estimate $P(w|k)$, $P(w|a)$, and $P(w|j)$, we consider an artificial document for each keyword k, author a and journal j corresponding to the concatenation of the abstracts of the documents where k, a and j occur, respectively. Since a document may contain more than one keyword k_i and one author a_j, we define $P(w|k)$ and $P(w|a)$ as:

$$P(w|k) = \frac{1}{K}\sum_{i=1}^{K} P(w|k_i) \tag{12}$$

$$P(w|a) = \frac{1}{A}\sum_{j=1}^{A} P(w|a_j) \tag{13}$$

where K and A are the number of keywords and authors in the document. The probabilities $P(w|j)$, $P(w|k_i)$ and $P(w|a_j)$ are estimated using maximum likelihood, analogously to $P(w|d)$. Alternatively, we can try to reflect the bigger influence of the first author by giving a higher weight γ to his probabilities. In that case, if there is more than one author, (13) becomes:

$$P(w|a) = \gamma P(w|a_1) + \frac{1-\gamma}{A-1}\sum_{j=2}^{A} P(w|a_j) \tag{14}$$

4.2.1. Latent Dirichlet Allocation

Two conceptually related abstracts may contain different terms (e.g. synonyms, misspellings, related terms), and may therefore not be recognized as similar. While this is a typical problem in information retrieval, it is aggravated here due to the short length of abstracts. To cope with this, methods can be used that recognize which *topics* are covered by an abstract, where topics are broader than keywords, but are still sufficiently discriminative to yield a meaningful description of the content of an abstract. This topical information is not directly available, but it can be estimated by using Latent Dirichlet Allocation (LDA) [2].

The idea behind LDA is that documents are generated by a (latent) set of topics, which are modeled as a probability distribution over terms. To generate a document, a distribution over those topics is set, and then, to generate each term w in the document, a topic z is sampled from the topic distribution, and w is sampled from the term distribution of the selected topic. In other words, the set of distributions ϕ over the terms in the collection (for each topic) and the set of distributions θ over all the topics (for each paper) need to be estimated. To do so, we use LDA with Gibbs sampling [13]. These probabilities are then estimated as:

$$P(w|z) = \hat{\phi}_z^{(w)} = \frac{n_z^{(w)} + \beta}{n_z^{(\cdot)} + W\beta} \tag{15}$$

$$P(z|\tau) = \hat{\theta}_z^{(d)} = \frac{n_z^{(d)} + \alpha}{n_\cdot^{(d)} + T\alpha} \tag{16}$$

where τ is the LDA model obtained with Gibbs sampling, W is the number of terms in the collection, and T is the number of topics. Parameters α and β intuitively specify how close (15) and (16) are to a maximum likelihood estimation: if their value is zero, (15) and (16) become a maximum likelihood estimation, while high values make them tend to a uniform distribution. The value $n_z^{(w)}$ is the number of times term w is assumed to have been generated by topic z, while $n_z^{(d)}$ is the number of times a term instance of document d is assumed to have been generated by topic z. Finally, $n_z^{(\cdot)}$ is the total number of times a term has supposedly been generated by topic z, and $n_\cdot^{(d)}$ is the total number of term instances of document d generated by any topic. All these values, except $n_\cdot^{(d)}$, which is simply the length of d, are unknown a priori.

The idea of the Gibbs sampling algorithm is to sample all variables from their distribution when conditioned on the current values of the rest of the variables. If repeated, the values will start to converge to the actual distribution. To apply the LDA algorithm, we first initialize it by randomly sampling a topic from a uniform distribution, for each occurrence of a term in every document; the topic is assigned as the generator of that instance of the term. By doing this, the aforementioned counts $n_z^{(w)}$, $n_z^{(d)}$ and $n_z^{(\cdot)}$ are randomly initialized. Then, an iterative process begins. In each iteration, for each instance w of a term in the collection, a topic is sampled based on probability estimates derived from the current assignments, i.e. the probability that topic z is chosen is given by

$$P(z|w,\tau) \propto P(w|z) \times P(z|\tau) = \frac{n_z'^{(w)} + \beta}{n_z'^{(\cdot)} + W\beta} \cdot \frac{n_z'^{(d)} + \alpha}{n'^{(d)}_{\cdot} + T\alpha} \tag{17}$$

where $n_z'^{(w)}$, $n_z'^{(d)}$, $n_z'^{(\cdot)}$ and $n'^{(d)}_{\cdot}$ are analogous to $n_z^{(w)}$, $n_z^{(d)}$, $n_z^{(\cdot)}$ and $n^{(d)}_{\cdot}$ respectively, but without including the current assignment of w. When the values converge, or, alternatively, after a given number of iterations, the algorithm stops, and ϕ and θ can finally be estimated according to (15) and (16).

Author information can also be used in an analogous way, as similar papers are often written by authors within the same community. This information about different communities may be useful, as it complements the information about topics: in the same way that a group of keywords can define a topic, a group of authors (a community) can define the set of topics they usually write about. By using the author information instead of the keywords, the underlying communities can be found. A community model thus becomes available by straightforwardly modifying Eqs. (15)–(17) as follows:

$$P(w|q) = \hat{\phi}_q^{(w)} = \frac{n_q^{(w)} + \beta}{n_q^{(\cdot)} + W\beta} \tag{18}$$

$$P(q|\kappa) = \hat{\theta}_q^{(d)} = \frac{n_q^{(d)} + \alpha}{n^{(d)}_{\cdot} + C\alpha} \tag{19}$$

$$P(q|w,\kappa) \propto P(w|q) \times P(q|\kappa) = \frac{n_q'^{(w)} + \beta}{n_q'^{(\cdot)} + W\beta} \cdot \frac{n_q'^{(d)} + \alpha}{n'^{(d)}_{\cdot} + C\alpha} \tag{20}$$

where C is the number of communities, q is a given community and κ is the new LDA model obtained with Gibbs sampling. The counts $n_q^{(w)}$, $n_q^{(d)}$, $n_q^{(\cdot)}$, $n_q'^{(w)}$, $n_q'^{(d)}$ and $n_q'^{(\cdot)}$ are defined analogously to $n_z^{(w)}$, $n_z^{(d)}$, $n_z^{(\cdot)}$, $n_z'^{(w)}$, $n_z'^{(d)}$ and $n_z'^{(\cdot)}$.

To find the underlying topics and communities, the LDA algorithm needs some input, namely the number T of topics and the number C of communities to be found. In Section 5.2 we study and discuss the best values for T and C. The topics and communities that are obtained from LDA can be used to improve the language model of a given document d. In particular, we propose to add $P(w|\tau)$ and $P(w|\kappa)$ to the right-hand side of (11), with the appropriate weights λ_i:

$$D(w) = \lambda_1 P(w|d) + \lambda_2 P(w|k) + \lambda_3 P(w|a) + \lambda_4 P(w|j) + \lambda_5 P(w|\tau) + \lambda_6 P(w|\kappa) + \lambda_7 P(w|\mathcal{C}) \tag{21}$$

$P(w|\tau)$ reflects the probability that term w is generated by the topics underlying document d. It can be estimated by considering that:

$$P(w|\tau) = \sum_{i=1}^{T} P(w|z_i) \times P(z_i|\tau) \tag{22}$$

On the other hand, $P(w|\kappa)$ represents the probability that term w is generated by the underlying communities, and is defined by:

$$P(w|\kappa) = \sum_{i=1}^{C} P(w|q_i) \times P(q_i|\kappa) \tag{23}$$

In summary, we can now build a model D for each document d interpolating not only some features as in (11), but also underlying information such as topics and communities as defined in (21). This method is further referred to as *LM0*.

4.3. Enriched estimations

Eq. (16) estimates the probability $P(z|\tau)$ of a topic z given an LDA model τ. However, this estimation is only based on the abstracts, while intuitively both authors and journals have an important influence on the topics: authors usually write on the same topics, and journals cover a more or less well-defined spectrum. Therefore, we propose to use both features in order to compute the estimations of ϕ and θ.

While the outline of the method remains the same, we rewrite (16) as

$$P(z|\tau) = \hat{\theta}_z^{(d)} = \frac{n_z^{(d)} + \alpha_1 + n_z^{(j)}\alpha_2 + n_z^{(a)}\alpha_3}{n^{(d)}_{\cdot} + T\alpha_1 + n^{(j)}_{\cdot}\alpha_2 + n^{(a)}_{\cdot}\alpha_3} \tag{24}$$

where new counts are introduced: $n_z^{(j)}$ is the number of times a term of journal j has been assigned to topic z, $n_\cdot^{(j)}$ is the total of terms (instances) of j, $n_z^{(a)}$ is the number of times a term of author a has been assigned to topic z, and $n_\cdot^{(a)}$ is the total of terms of a. Also, the value of α in (16) is now split into α_1, α_2 and α_3. These values, which control the importance of each feature in the smoothing method, are discussed in Section 5.2. This modification implies changes in the Gibbs sampling algorithm as well, replacing the part of the estimation of $P(z|\tau)$ in (17) by (24). We call this method *LM0e*.

4.4. Improved initialization

A different approach to improve *LM0* is by taking advantage of the fact that keywords have been assigned to each paper. In particular, we propose to exploit the available keywords to improve the initialization part of the Gibbs sampling algorithm, and therefore to get more accurate estimations.

The idea is to cluster the keywords and identify each cluster with a topic. The parameters of the multinomial distributions corresponding with each topic can then initially be estimated from these clusters. Conceptually, we represent each keyword by an artificial document, corresponding to the concatenation of the abstracts of the papers to which that keyword has been assigned (analogously to the d_k documents in Section 3.2). Similarity between keywords is then estimated by calculating the cosine similarity between the corresponding artificial documents. The clusters are then obtained using the *K*-means algorithm [23]. This is a basic clustering algorithm, but also fast, well known, and easy to implement. However, other algorithms can be considered.

Once the clusters have been determined, we represent them by the concatenation of all abstracts to which at least one of the keywords in the clusters was assigned. We can then estimate a multinomial distribution from these documents, and initialize the Gibbs sampling procedure with it.

For this process, only the keywords which appear in a minimum number of documents are used (the value of this threshold is discussed in Section 5.2). This means that the terms occurring in documents that do not contain any of those keywords are not taken into account to build the clusters (and therefore, to compute the initial values for the parameters). Also, there is no information about the topics that generate the terms which occur exclusively in those documents. In those cases, the topic is not sampled from the resulting multinomial distributions, but from a uniform distribution, i.e., we fall back on the basic initialization method that is usually considered. For more details, we refer to the example in Section 4.5, and in particular to Section 4.5.4.

In addition to terms that do not occur in the artificial cluster documents at all, we may also consider that terms that are rare in these clusters may need to be smoothed. Initial experiments, however, indicated that this does not actually improve the performance, hence we will not consider this additional form of smoothing in our experiments, avoiding the unnecessary introduction of more parameters.

Once initial values for all the parameters in (15) and (16) have been set, we can apply the iterative part of the LDA algorithm. We call this method *LM2*.[2] By clustering authors instead of keywords, the community models are analogously improved in this method.

This idea can also be used to improve *LM0e*. In that case, the counts $n_z^{(j)}, n_\cdot^{(j)}, n_z^{(a)}$ and $n_\cdot^{(a)}$ must be initialized as well. After sampling a topic for each instance of a term in a given document d, the respective counts for the journal corresponding to that document are increased. In order to estimate which author generated a term instance, a uniform distribution on the total number of authors of d is used, and the counts corresponding to the author sampled from it are increased. The rest of the process is analogous to method *LM2*; we call this method *LM2e*.

4.5. Running example

We provide an example of how the proposed method based on language models works as a whole, step by step. In particular, we detail how the language models are created and interpolated, how the LDA step works, and how the initialization of LDA can be improved.

We consider the following collection $\mathcal{C}$ consisting of four documents. In order to improve readability, we use letters instead of words, keywords, authors' names, or journals:

$$d_1 = \{abs = (a,b,a,c,d,a), kws = (k_1,k_2), aut = (u_1,u_2), jou = (j_1)\}$$
$$d_2 = \{abs = (a,a,d,a,b,a), kws = (k_1,k_3), aut = (u_1,u_3), jou = (j_1)\}$$
$$d_3 = \{abs = (a,b,a), kws = (k_2,k_3), aut = (u_2,u_4), jou = (j_2)\}$$
$$d_4 = \{abs = (a,b,b,e), kws = (k_4), aut = (u_5,u_6), jou = (j_2)\}$$

4.5.1. Step 1: basic language models

As explained in Section 4.1, the probabilities are initially only based on the abstract information and estimated using maximum likelihood. In this way, the probabilities of a term being generated by the language model of d_1 are:

[2] In previous work [16], we studied a third method called LM1, which we do not consider in this paper. To avoid confusion we have decided to keep the same notation, and therefore we speak about method LM2 without having mentioned a method LM1 before.

$$P(a|d_1)=\frac{3}{6} \quad P(b|d_1)=\frac{1}{6} \quad P(c|d_1)=\frac{1}{6}$$
$$P(d|d_1)=\frac{1}{6} \quad P(e|d_1)=0$$

Also, the probabilities of a term being generated by the background model must be estimated:

$$P(a|\mathcal{C})=\frac{10}{19} \quad P(b|\mathcal{C})=\frac{5}{19} \quad P(c|\mathcal{C})=\frac{1}{19}$$
$$P(d|\mathcal{C})=\frac{2}{19} \quad P(e|\mathcal{C})=\frac{1}{19}$$

Now the basic model D_1 can be calculated for d_1 (models D_2, D_3 and D_4 are built analogously):

$$D_1(a)=\lambda\frac{3}{6}+(1-\lambda)\frac{10}{19} \quad D_1(b)=\lambda\frac{1}{6}+(1-\lambda)\frac{5}{19}$$
$$D_1(c)=\lambda\frac{1}{6}+(1-\lambda)\frac{1}{19} \quad D_1(d)=\lambda\frac{1}{6}+(1-\lambda)\frac{2}{19}$$
$$D_1(e)=\lambda 0+(1-\lambda)\frac{1}{19}$$

4.5.2. Step 2: interpolated language models

However, as proposed in Section 4.2, we do not only want to use the abstract, but also the other features. For example, to use the keyword information, we first consider an artificial document for each keyword in the collection. This artificial document contains a concatenation of the abstracts of those documents where the keyword occurs:

$$k_1=\{a,b,a,c,d,a,a,a,d,a,b,a\}$$
$$k_2=\{a,b,a,c,d,a,a,b,a\}$$
$$k_3=\{a,a,d,a,b,a,a,b,a\}$$
$$k_4=\{a,b,b,e\}$$

The probabilities can now be estimated similarly to the case of the abstracts. For k_1, for instance:

$$P(a|k_1)=\frac{7}{12} \quad P(b|k_1)=\frac{2}{12} \quad P(c|k_1)=\frac{1}{12}$$
$$P(d|k_1)=\frac{2}{12} \quad P(e|k_1)=0$$

The same is done for k_2, k_3 and k_4. The same process is repeated for the authors and the journal: an artificial document is considered for each author (resp. journal) in the collection, and then the probabilities can be estimated. After doing this, new models can be calculated with the new probabilities, as is done in Eq. (11). Some examples:

$$D_1(a)=\lambda_1\frac{3}{6}+\lambda_2\frac{\frac{7}{12}+\frac{5}{9}}{2}+\lambda_3\frac{\frac{7}{12}+\frac{5}{9}}{2}+\lambda_4\frac{7}{12}+\lambda_5\frac{10}{19}$$
$$D_3(a)=\lambda_1\frac{2}{3}+\lambda_2\frac{\frac{5}{9}+\frac{6}{9}}{2}+\lambda_3\frac{\frac{5}{9}+\frac{2}{3}}{2}+\lambda_4\frac{3}{7}+\lambda_5\frac{10}{19}$$
$$D_1(c)=\lambda_1\frac{1}{6}+\lambda_2\frac{\frac{1}{12}+\frac{1}{9}}{2}+\lambda_3\frac{\frac{1}{12}+\frac{1}{9}}{2}+\lambda_4\frac{7}{12}+\lambda_5\frac{1}{19}$$

It can be seen that, in the case of keywords and authors, the final probability is estimated by calculating the average of the probabilities of the keywords (resp. authors) that occur in that document.

4.5.3. Step 3: Latent Dirichlet Allocation

In Section 4.2.1 we have proposed using LDA in order to extract new information, this time regarding the (underlying) topics. First, the number of topics to be found must be given. In this example we assume that there are two underlying topics, *A* and *B*. Then, as explained in Section 4.2.1, we need some counts to estimate the required probabilities (15) and (16). These counts are initialized this way: for each term w in the abstract of each document d, a topic z is randomly sampled. This topic is then assumed to have generated that very instance of the term, which means that the counts $n_z^{(w)}$, $n_z^{(d)}$ and $n_z^{(\cdot)}$ are increased. By doing so, we obtain for example:

$$n_A^{(a)}=7 \quad n_A^{(b)}=3 \quad n_A^{(c)}=0 \quad n_A^{(d)}=1 \quad n_A^{(e)}=0$$
$$n_B^{(a)}=3 \quad n_B^{(b)}=2 \quad n_B^{(c)}=1 \quad n_B^{(d)}=1 \quad n_B^{(e)}=1$$

$$n_A^{(d_1)} = 4 \quad n_A^{(d_2)} = 3 \quad n_A^{(d_3)} = 3 \quad n_A^{(d_4)} = 1$$
$$n_B^{(d_1)} = 2 \quad n_B^{(d_2)} = 3 \quad n_B^{(d_3)} = 0 \quad n_B^{(d_4)} = 3$$

$n_A^{(\cdot)} = 11$, the total number of instances generated by topic A
$n_B^{(\cdot)} = 8$, the total number of instances generated by topic B
These values are then used to initialize the LDA algorithm. For example, for term a and document d_1 we obtain:

$$\hat{\phi}_A^{(a)} = \frac{7+\beta}{11+5\beta} \quad \hat{\phi}_B^{(a)} = \frac{3+\beta}{8+5\beta}$$
$$\hat{\theta}_A^{(d_1)} = \frac{4+\alpha}{6+2\alpha} \quad \hat{\theta}_B^{(d_1)} = \frac{2+\alpha}{6+2\alpha}$$

The LDA algorithm can now be run. In this example we set the parameters $\alpha = 0.16$ and $\beta = 0.1$, and the following estimations for the desired probabilities are obtained:

$$\hat{\phi}_A^{(a)} = 0.93 \quad \hat{\phi}_A^{(b)} = 0.018 \quad \hat{\phi}_A^{(c)} = 0.018 \quad \hat{\phi}_A^{(d)} = 0.018 \quad \hat{\phi}_A^{(e)} = 0.018$$
$$\hat{\phi}_B^{(a)} = 0.35 \quad \hat{\phi}_B^{(b)} = 0.35 \quad \hat{\phi}_B^{(c)} = 0.076 \quad \hat{\phi}_B^{(d)} = 0.15 \quad \hat{\phi}_B^{(e)} = 0.076$$

$$\hat{\theta}_A^{(d_1)} = 0.02 \quad \hat{\theta}_A^{(d_2)} = 0.5 \quad \hat{\theta}_A^{(d_3)} = 0.65 \quad \hat{\theta}_A^{(d_4)} = 0.037$$
$$\hat{\theta}_B^{(d_1)} = 0.97 \quad \hat{\theta}_B^{(d_2)} = 0.5 \quad \hat{\theta}_B^{(d_3)} = 0.35 \quad \hat{\theta}_B^{(d_4)} = 0.96$$

With these values, and following (22), we can calculate the probability of a given term being generated by a given topic, and then add that probability to the document model as shown in (21). For example:

$$D_1(a) = \lambda_1 \frac{3}{6} + \lambda_2 \frac{\frac{7}{12}+\frac{5}{9}}{2} + \lambda_3 \frac{\frac{7}{12}+\frac{5}{9}}{2} + \lambda_4 \frac{7}{12} + \lambda_5 (0.93 \times 0.02 + 0.35 \times 0.97) + \lambda_6 \frac{10}{19}$$
$$D_3(a) = \lambda_1 \frac{2}{3} + \lambda_2 \frac{\frac{5}{9}+\frac{6}{9}}{2} + \lambda_3 \frac{\frac{5}{9}+\frac{2}{3}}{2} + \lambda_4 \frac{3}{7} + \lambda_5 (0.93 \times 0.65 + 0.35 \times 0.35) + \lambda_6 \frac{10}{19}$$
$$D_1(c) = \lambda_1 \frac{1}{6} + \lambda_2 \frac{\frac{1}{12}+\frac{1}{9}}{2} + \lambda_3 \frac{\frac{1}{12}+\frac{1}{9}}{2} + \lambda_4 \frac{7}{12} + \lambda_5 (0.018 \times 0.02 + 0.076 \times 0.97) + \lambda_6 \frac{1}{19}$$

The same process is followed to use the information about the communities. For the sake of simplicity, we do not consider them here, and therefore the term regarding the communities in the previous examples for $D_1(a)$, $D_3(a)$ and $D_1(c)$ is missing.

4.5.4. Step 4: LDA improvements

Sections 4.3 and 4.4 propose how to improve the previously explained method. To enrich the estimations new variables are introduced in Eq. (24). In order to use these variables, we consider artificial documents as in Section 4.5.2, and then we use them to initialize the variables as in the previous section. Since there are no other differences, we do not go into details here.

The use of the improved initialization does require a more detailed example. First, as explained in Section 4.4, the keywords are clustered. Only those keywords which occur in a minimum number of clusters are used. In this example we set the minimum to 2. After clustering the keywords, suppose two clusters A and B are obtained:

$$A = \{k_1, k_2\}$$
$$B = \{k_3\}$$

with their respective artificial documents c_A and c_B:

$$c_A = \{a, b, a, c, d, a, a, a, d, a, b, a, a, b, a\}$$
$$c_B = \{a, a, d, a, b, a, a, b, a\}$$

According to the information in the clusters, topic A has generated term a nine times, and B has generated a six times. This information leads to the initial estimation $P(a|A) = 9/15$ and $P(a|B) = 6/15$. Then, to estimate which topic actually generates a specific instance of a in document d_1, we just sample the topic from that distribution. If the sampled topic is, for example, A, we increase the counts $n_A^{(a)}, n_A^{(d_1)}$, and $n_A^{(\cdot)}$. The process is analogous for the remaining occurrences of a, b, c and d. However, term e does not occur in the artificial cluster documents, and therefore there is no information about it. To estimate the topic which generates e, we use a uniform distribution on the T topics, i.e. $P(e|A) = 1/2$ and $P(e|B) = 1/2$. This leads us to the following initial values for those variables for example:

$$n_A^{(a)} = 8 \quad n_A^{(b)} = 3 \quad n_A^{(c)} = 1 \quad n_A^{(d)} = 2 \quad n_A^{(e)} = 0$$
$$n_B^{(a)} = 3 \quad n_B^{(b)} = 1 \quad n_B^{(c)} = 0 \quad n_B^{(d)} = 0 \quad n_B^{(e)} = 1$$

$$n_A^{(d_1)} = 5 \quad n_A^{(d_2)} = 4 \quad n_A^{(d_3)} = 3 \quad n_A^{(d_4)} = 2$$
$$n_B^{(d_1)} = 1 \quad n_B^{(d_2)} = 2 \quad n_B^{(d_3)} = 0 \quad n_B^{(d_4)} = 2$$

As we can see, if c only occurs in c_A, the only topic that can generate it should be A. However, with the random initialization, as shown in Section 4.5.3, it could be assumed to be generated by B. Of course, the execution of LDA can correct this later, but there is no guarantee about it. The improved initialization, on the other hand, already departs from more realistic/correct assumptions.

With both improvements, the rest of the LDA process remains the same. Finally, when the definitive models have been calculated, they can be compared by using (9).

5. Experimental evaluation

5.1. Experimental set-up

To build a test collection and evaluate the proposed methods, we downloaded a portion of the ISI Web of Science,[3] consisting of files with information about papers from 19 journals in the Artificial Intelligence domain. These files contain, among other data, the abstract, authors, journal, and keywords freely chosen by the authors. A total of 25,964 paper descriptions were retrieved, although our experiments are restricted to the 16,597 papers for which none of the considered fields is empty.

The ground truth for our experiments is based on annotations made by 8 experts.[4] First, 220 documents were selected, and each of them was assigned to an expert sufficiently familiar with it. Then, using tf–idf with cosine similarity, the 30 most similar papers in the test collection were found for each of the 220 papers. Each of those 30 papers was manually tagged by the expert as either similar or dissimilar. To evaluate the performance of the methods, each paper $\mathbf{p}$ is thus compared against 30 others,[5] some of which are tagged as similar. The approaches for assessing paper similarity discussed in Sections 3 and 4 can then be used to rank the 30 papers, such that ideally the papers similar to $\mathbf{p}$ appear at the top of the ranking. In principle, we thus obtain 220 rankings per method. However, due to the fact that some of the lists contained only dissimilar papers, and that sometimes the experts were not certain about the similarity of some items, the initial 220-paper set was reduced to 209 rankings. To evaluate these rankings, we use two well-known measures:

- *Mean Average Precision (MAP).* This measure takes into account the position of every hit within the ranking, and is defined by:

$$MAP = \frac{\sum_{r=1}^{|R|} AvPrec(r)}{|R|} \tag{25}$$

  where $|R|$ is the total number of rankings and *AvPrec* is the average precision of a ranking, defined by:

$$AvPrec = \frac{\sum_{i=1}^{n} Prec(i) \times rel(i)}{number\ of\ relevant\ documents} \tag{26}$$

  with $Prec(i)$ the precision at cut-off i in the ranking (i.e. the percentage of the i first ranked items that are relevant) and $rel(i) = 1$ if the item at rank i is a relevant document ($rel(i) = 0$ otherwise).
- *Mean Reciprocal Rank (MRR).* Unlike MAP, this measure only takes into account the first hit within the rankings, along with its position. It is defined by:

$$MRR = \frac{\sum_{r=1}^{|R|} RR(r)}{|R|} \tag{27}$$

  where *RR* is the reciprocal rank of a ranking:

$$RR = \frac{1}{fhit} \tag{28}$$

  with *fhit* the rank of the first hit in the ranking.

5.2. Results

5.2.1. Vector space model

Table 1 summarizes the results of the experiment for the approaches based on the vector space model, as described in Section 3. As already mentioned in that section, the vectors representing the documents are compared using the following

[3] http://apps.isiknowledge.com.

[4] The set of annotations is publicly available at http://www.cwi.ugent.be/respapersim/.

[5] During the annotation process it was also possible to tag some items as "Don't know" for those cases where the expert had no certainty about the similarity. These items are ignored and therefore some papers are compared to less than 30 others.

measures: cosine (*cos*), generalized Jaccard (*g.jacc*), extended Jaccard (*e.jacc*), and Dice (*dice*) similarity. In this table it is interesting to observe that the *abstract* method, traditionally combined with cosine similarity, performs significantly better when combined with the general Jaccard similarity measure (paired t-test, $p < 0.001$). This is the reason why we show double results for the *ESA* methods: the second block of the table summarizes the results obtained by building the $\mathbf{d_E}$ vectors using cosine similarity as proposed in Section 3.2 and defined in (7), while the third block substitutes the cosine similarity for general Jaccard. However, although the results in this last case are slightly better when the resulting $\mathbf{d_E}$ vectors are compared using cosine similarity, when comparing with the other similarity measures the results are slightly worse. On the other hand, neither *ESA-kws* nor *ESA-aut* can outperform *abstract*, despite using two features (abstract and keywords/authors) instead of just one as *abstract* does. It turns out that the journal information is too general, hence the especially bad performance of *ESA-jou*.

5.2.2. Language modeling

Table 2 shows the results obtained with the language modeling methods, described in Section 4. The λ configurations in the first columns correspond to those controlling the weight of abstract, keywords, authors, journal, topics, and communities, in that order (see Table 3).

The first block of the table summarizes the results obtained with language models that only use one of the features. We find that language models which only use the abstract (line 1) significantly improve the performance of most of the vector space methods (paired t-test, $p < 0.001$), the only exception being when general Jaccard is used to compare the abstracts ($p \simeq 0.089$). Models uniquely based on other features can perform slightly better than *abstract* (depending on the chosen similarity measure used by the latter), but these improvements were not found to be significant. However, these results are still useful as an indication of the amount of information contained in each of the features: language models based exclusively on keywords or on authors perform comparable to the method *abstract*. Using only topics yields such results when *LM2e* is used, while using communities performs slightly worse. The information contained in the journal feature is clearly poorer. Moreover, Fig. 2 shows that giving a higher weight to the first author when modeling a paper, as proposed in Section 4.2, does not make a big difference.

In the second block of Table 2 we examine different combinations of two features: abstract with topics on lines 7–9, and abstract with keywords on lines 10–12. These results confirm that the abstract contains the most information, and should be assigned a high weight. On the other hand, we can observe how the topics, when combined with the abstract, yield a better MAP score. In particular, the MAP scores on line 9 are significantly better than those on line 12 (*LM0*: $p \simeq 0.003$; *LM2*: $p \simeq 0.001$; *LM2e*: $p < 0.001$). The differences are also significant between lines 8 and 11 for *LM2* and *LM2e* (*LM0*: $p \simeq 0.062$; *LM2*: $p \simeq 0.005$; *LM2e*: $p < 0.001$), and between lines 7 and 10 for *LM2e* (*LM2e*: $p \simeq 0.026$). Other combinations of two features perform worse.

The third block shows the results of combining abstract and topics, with keywords, authors, and journal. It is clear that giving a small weight to keywords is beneficial, as it leads to high scores, which are significantly better than the configurations in lines 10–12 ($p < 0.001$ for the three methods *LM0*, *LM2* and *LM2e*). For methods *LM0* and *LM2*, the improvement is significant with respect to the configurations in lines 7–9 as well ($p < 0.029$, resp. $p < 0.03$). Using authors and journal also means an improvement, but smaller than that achieved with the keywords. Combining more than three features, as in lines

Table 1
Results obtained with the approaches based on the vector space model (methods described in Section 3). The bold values indicate the highest MAP and MRR values.

	cos	*dice*	*e.jacc*	*g.jacc*
MAP				
Abstract	0.546	0.546	0.546	**0.604**
Keywords	0.497	0.5	0.5	0.486
ESA-kws (cos)	0.576	0.549	0.549	0.529
ESA-aut (cos)	0.576	0.563	0.563	0.537
ESA-jou (cos)	0.397	0.404	0.404	0.329
ESA-kws (g.jacc)	0.599	0.536	0.536	0.504
ESA-aut (g.jacc)	0.582	0.553	0.553	0.512
ESA-jou (g.jacc)	0.403	0.37	0.37	0.273
MRR				
Abstract	0.726	0.726	0.726	**0.779**
Keywords	0.71	0.724	0.718	0.703
ESA-kws (cos)	0.738	0.704	0.704	0.701
ESA-aut (cos)	0.744	0.715	0.715	0.704
ESA-jou (cos)	0.546	0.554	0.554	0.42
ESA-kws (g.jacc)	0.749	0.72	0.72	0.695
ESA-aut (g.jacc)	0.736	0.736	0.736	0.697
ESA-jou (g.jacc)	0.565	0.524	0.524	0.32

Table 2
Results obtained with the approaches based on language modeling (methods described in Section 4). The bold values indicate the highest MAP and MRR values.

Line	λ-configuration						MAP			MRR		
	abs	kws	aut	jou	tpc	com	LM0	LM2	LM2e	LM0	LM2	LM2e
1	0.9	0	0	0	0	0	0.622	0.622	0.622	0.791	0.791	0.791
2	0	0.9	0	0	0	0	0.558	0.558	0.558	0.73	0.73	0.73
3	0	0	0.9	0	0	0	0.557	0.557	0.557	0.711	0.711	0.711
4	0	0	0	0.9	0	0	0.314	0.314	0.314	0.382	0.382	0.382
5	0	0	0	0	0.9	0	0.505	0.523	0.585	0.655	0.674	0.751
6	0	0	0	0	0	0.9	0.491	0.491	0.491	0.621	0.621	0.621
7	0.7	0	0	0	0.2	0	0.642	0.647	0.655	0.805	0.798	0.82
8	0.2	0	0	0	0.7	0	0.607	0.625	0.644	0.774	0.775	0.814
9	0.45	0	0	0	0.45	0	0.648	0.655	0.66	0.816	0.804	0.819
10	0.7	0.2	0	0	0	0	0.625	0.625	0.625	0.793	0.793	0.793
11	0.2	0.7	0	0	0	0	0.574	0.574	0.574	0.746	0.746	0.746
12	0.45	0.45	0	0	0	0	0.597	0.597	0.597	0.773	0.773	0.773
13	0.4	0.1	0	0	0.4	0	0.671	0.681	0.678	0.822	0.823	0.824
14	0.1	0.4	0	0	0.4	0	0.61	0.611	0.624	0.776	0.773	0.783
15	0.4	0.4	0	0	0.1	0	0.612	0.616	0.619	0.777	0.781	0.785
16	0.3	0.3	0	0	0.3	0	0.632	0.641	0.648	0.791	0.797	0.802
17	0.4	0	0.1	0	0.4	0	0.66	0.67	0.66	0.812	0.805	0.806
18	0.4	0	0	0.1	0.4	0	0.649	0.655	0.665	0.801	0.802	0.818
19	0.3	0.1	0.1	0.1	0.3	0	0.667	0.675	0.67	0.81	0.812	0.818
20	0.4	0.1	0.1	0	0.3	0	0.667	0.675	0.668	0.812	0.819	0.816
21	0.4	0.1	0	0.1	0.3	0	0.674	0.681	0.683	0.826	0.82	0.828
22	0.4	0	0	0	0.4	0.1	0.647	0.656	0.666	0.803	0.805	0.827
23	0.3	0.1	0	0	0.3	0.2	0.68	0.685	**0.687**	0.822	0.827	**0.831**
24	0.4	0.1	0.1	0	0.3	0.1	0.673	0.68	0.683	0.822	0.822	**0.831**
25	0.4	0.1	0	0.1	0.3	0.05	0.675	0.684	0.678	0.823	0.824	0.82

Table 3
Configurations for the study of the impact of the first author's weight.

	abs	*kws*	*aut*	*jou*	*tpc*	*com*
cfg.1	0	0	0.9	0	0	0
cfg.2	0.3	0.1	0.1	0.1	0.3	0
cfg.3	0.4	0.1	0.1	0	0.3	0
cfg.4	0.3	0.3	0.3	0	0	0

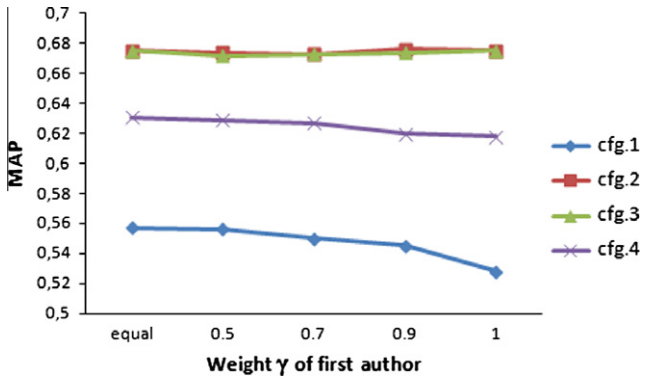

Fig. 2. Impact of the first author's weight (configuration values shown in Table 3).

19–21, does not show a significant improvement with respect to the previous lines. In Fig. 3 we further explore the importance of the abstract and the topics. We set the weight of the keywords to a fixed value of 0.1, and the remaining weight of 0.8 is divided between abstract and topics. What is particularly noticeable is that ignoring the abstract is penalized stronger than ignoring the topics, but the optimal performance is obtained when both features are given approximately the same weight.

Finally, in the fourth and last block we also include the communities. Since abstracts and topics have proven to contain most of the information, they still get higher weights. However, by assigning a small weight to the communities, we can achieve the highest scores (although the difference with the best scores in the third block is not significant).

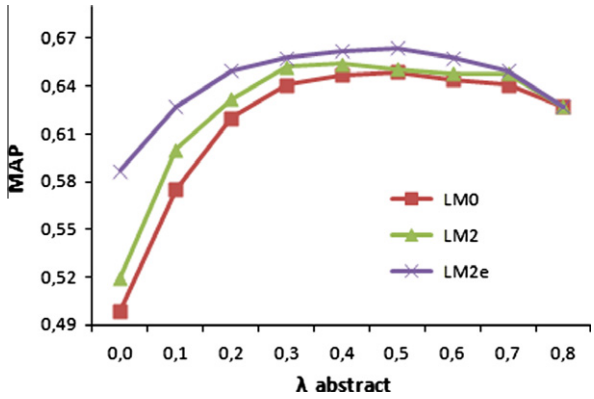

Fig. 3. Importance of abstract vs. topics.

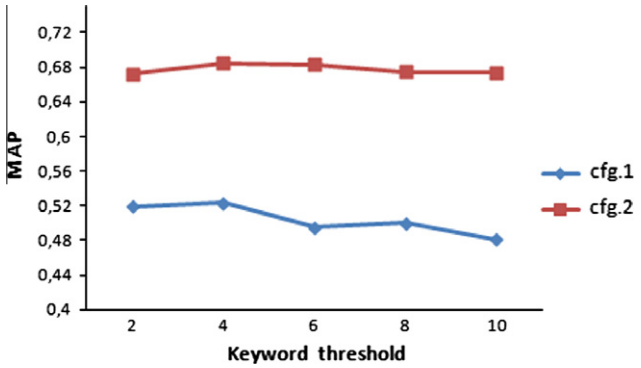

Fig. 4. Impact of the keyword threshold, with *cfg*.1: $\lambda_{tpc} = 0.9$ and *cfg*.2: $\lambda_{abs} = 0.3$, $\lambda_{kws} = 0.1$, $\lambda_{tpc} = 0.3$, $\lambda_{com} = 0.2$.

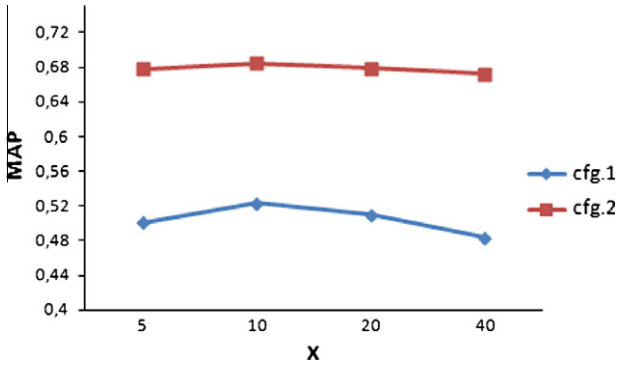

Fig. 5. Impact of the number T of topics, $T = kws/X$, with *cfg*.1: $\lambda_{tpc} = 0.9$ and *cfg*.2: $\lambda_{abs} = 0.3$, $\lambda_{kws} = 0.1$, $\lambda_{tpc} = 0.3$, $\lambda_{com} = 0.2$.

We can note that *LM2* only slightly improves *LM0*, but larger differences in MAP scores can be observed between *LM0* and *LM2e* in those cases in which the topics are given more importance, such as in line 8 ($p \simeq 0.001$). The difference is particularly striking when only the topics are used to create the models (line 5, with $\lambda_{topics} = 0.9$, $p < 0.001$), which shows how much LDA can benefit from the initialization based on the different features.

5.2.3. Parameter tuning

For the experiments concerning the language modeling methods, we fixed the sum of these weights to 0.9, and set the general smoothing factor (λ_7 in (21)) to 0.1. Also, the threshold determining the minimum number of documents in which a keyword must appear in order to be taken into account for the clusters was fixed to 4. This means that a total number of 3219 keywords was used. The reason for this choice lies mainly in computing performance constraints, but also in the fact

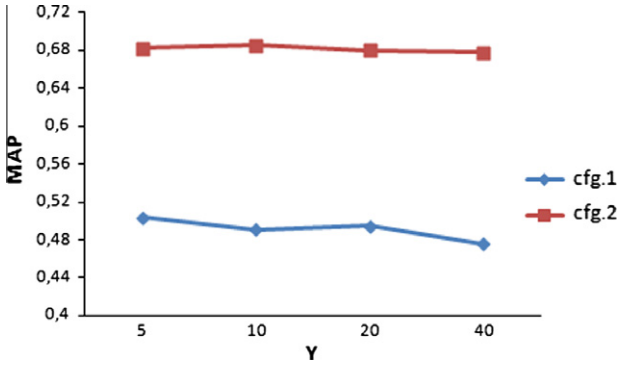

Fig. 6. Importance of the number C of communities, $C = authors/Y$, with *cfg*.1: $\lambda_{com} = 0.9$ and *cfg*.2: $\lambda_{abs} = 0.3$, $\lambda_{kws} = 0.1$, $\lambda_{tpc} = 0.3$, $\lambda_{com} = 0.2$.

that keywords appearing in just a couple of documents may introduce noise. The choice of the number of keywords influences the number T of topics, since we fixed this number to 10% of the number of keywords. Therefore, the results displayed in Table 2 were obtained with 321 topics. Figs. 4 and 5, however, show the limited importance of these choices with respect to the final results. Furthermore, parameters α and β introduced in (15) and (16) are fixed to $\alpha = 50/T$ (i.e. $\alpha = 0.157$ in this case) and $\beta = 0.1$, since these are the values typically used for LDA with Gibbs sampling. Finally, the communities used in our experiments (line 6 and last block of Table 2) were calculated with method *LM2* and a fixed number C of communities equal to 201. This value of C was obtained analogously to T: 2017 authors occurred in more than four documents and then we divided by 10. In Fig. 6, however, we can observe the robustness of the method w.r.t. the choice of the value of C.

As for method *LM2e*, the chosen values for the α-weights in these experiments are $\alpha_1 = 0.8\alpha$ and $\alpha_3 = 0.2\alpha$. In other words, the author information is now added to the LDA smoothing with a mild weight. However, no weight is given to the journal, since preliminary experiments showed that the performance was not improved when using the journal information, as it was, like in *ESA-jou*, too general.

6. Conclusion and future work

We have proposed and compared several content-based methods to compare research paper abstracts. To do so, we have studied and enriched existing methods by taking advantage of the semi-structured information that is usually available in the description of a research paper: a list of keywords, a list of authors, and the journal in which it was published. These methods, based either on the vector space model or on language modeling, perform comparably when only the abstract is considered. However, when the additional document features are used, important differences are noticed. The proposed methods based on the vector space model cannot outperform the traditional method, although the ESA methods, which combine abstract with another feature, do outperform the standard tf–idf approach in the case where the popular cosine similarity is considered. In fact, our results suggest that cosine similarity is far from an optimal choice for assessing document similarity in the vector space model, at least in the case of research paper abstracts. Language models, however, have proven more suitable in this context than any of the vector space methods we considered, as the results show that they are able to take advantage of the extra document features. By interpolating models based on the different features, the typical approach where only the abstract is used is significantly improved. Finally, we have also explored how LDA could be used in this case to discover latent topics and communities, and a method has been proposed to effectively exploit the keywords and authors associated with a paper to significantly improve the performance of the standard LDA algorithm.

All experiments were performed with an annotated dataset which we have made publicly available. To our knowledge, we are the first to contribute such a public dataset to evaluate research paper similarity.

The present work leaves some issues open for future work, offering two main directions for further research. On the one hand, there are still some points in the studied methods that may be improved. The use of the author field is a good example. Author names in bibliographical databases are prone to problems due to several reasons: badly recorded names, the appearance of several variants of an author's name, or different authors having the same name are only some of them. This is a non-trivial problem that comprises several challenges [33] which we have not addressed here. Also, as mentioned in Section 4.4, alternative clustering algorithms could be used for the LDA initialization. Or, focusing on the vector space model approaches, it may be interesting to consider other approaches based in concept representation (similarly to ESA), such as the one proposed in [11].

On the other hand, a very interesting idea is to implement a scientific article recommender system in which the studied methods are applied. Such a system can build user profiles based on the previously published papers of each user, and/or on papers in which he has already expressed an interest, and then compare those papers with the rest of the papers in the database or databases used. Of course, such a system would have some of the limitations inherent to content-based systems, so a next step would be combining the proposed methods with other ideas such as collaborative filtering or the use of authoritativeness.

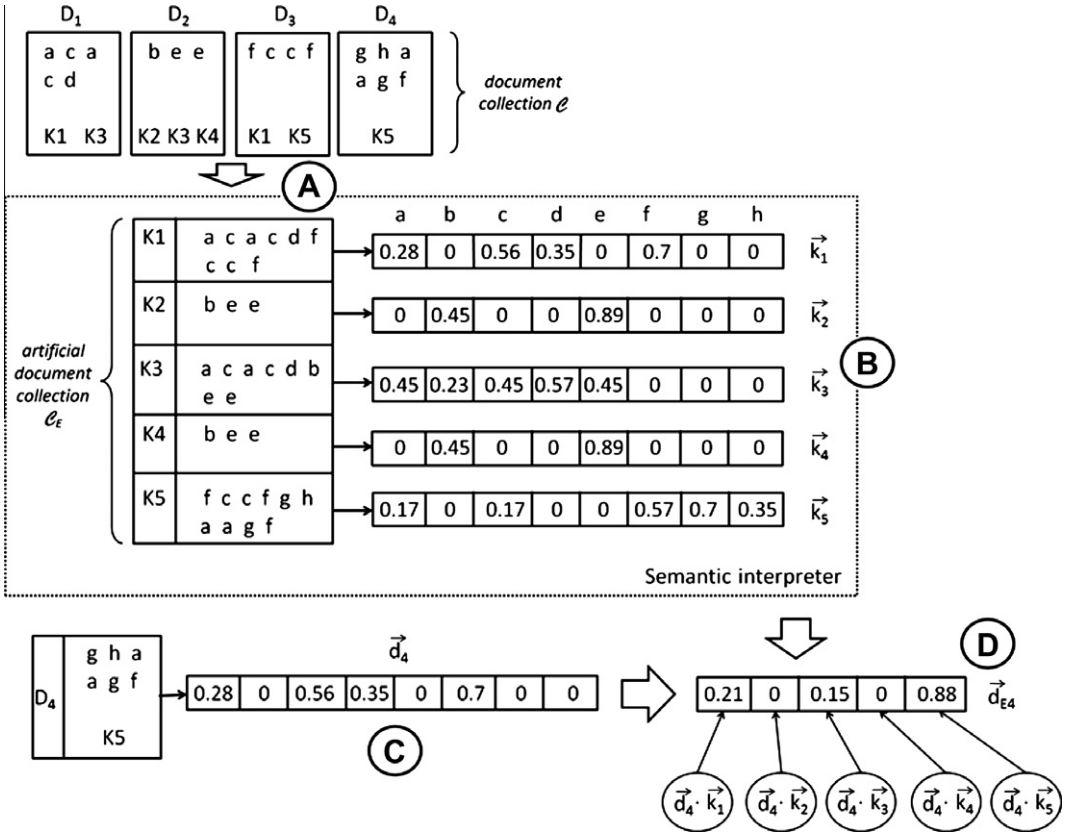

Fig. 7. How ESA vectors are calculated.

Appendix A. An ESA example

In this appendix we show a small example of how ESA exactly works. For the sake of simplicity, we have used letters instead of words in order to represent the terms. The example, depicted in Fig. 7, shows the whole process of calculating the ESA vectors for a collection $\mathcal{C}$ of four documents, $\mathcal{C} = \{D_1, D_2, D_3, D_4\}$.

First, an artificial document is considered for each keyword by concatenating the abstracts of the documents where they occur, forming a new collection $\mathcal{C}_E$ (A). Then, a weighted term vector is calculated for each of those documents using tf–idf (1), which is then normalized (B). Weighted term vectors are calculated analogously for each document in the original collection $\mathcal{C}$, which contains the documents that we want to represent as ESA vectors (C). Finally, each vector $\mathbf{d_i}$ resulting from step C is compared to every vector $\mathbf{k_i}$ resulting from step B. The result of each of those comparisons is used as the weight of the corresponding components in the resulting ESA vector $\mathbf{d_{Ei}}$ (D).

Appendix B. A detailed case study

This appendix offers a more qualitative view on the results, rather than the quantitative view offered in Section 5.2.2, in order to gain insight into the improvements of the proposed methods. To do so, we detail a particular case where the system must find matches for the following paper: *"(v, T)-fuzzy rough approximation operators and the TL-fuzzy rough ideals on a ring"*.[6]

As explained in Section 5.1, a paper is compared to 30 others tagged as similar or not similar, obtaining then a ranking where the most similar papers occur in the highest positions. Table B.4 shows the titles of the top ten papers of such a ranking when methods *abstract (g.jacc)*, *LM0* and *LM2e* are used to find matches for the aforementioned paper. The actual hits are highlighted in bold. Also, at the bottom of the table the average precision for each method is shown.

It can be seen that the top four positions for *LM2e* are indeed hits. *LM0* already misses one of those four hits (it appears at the 6th position), while *abstract* ranks its first hit in the 10th position. This is due to the fact that the abstracts of the hits, although they share some vocabulary with that of the given document, do not have so many (meaningful) terms in common

[6] Only the titles are used here; for information about the rest of features used by the system we refer to the articles' records in the ISI Web of Science.

Table B.4
Top ten matches for the studied paper.

Rank	*Abstract*	*LM0*	*LM2e*
1	On characterizations of (*I*, *T*)-fuzzy rough approximation operators	**Rough set theory applied to (fuzzy) ideal theory**	**Roughness in rings**
2	Generalized fuzzy rough approximation operators based on fuzzy coverings	**Roughness in rings**	**Generalized lower and upper approximations in a ring**
3	Constructive and axiomatic approaches of fuzzy approximation operators	**The product structure of fuzzy rough sets on a group and the rough T-fuzzy group**	**Rough set theory applied to (fuzzy) ideal theory**
4	The minimization of axiom sets characterizing generalized approximation operators	Generalized fuzzy rough approximation operators based on fuzzy coverings	**The product structure of fuzzy rough sets on a group and the rough T-fuzzy group**
5	Minimization of axiom sets on fuzzy approximation operators	An axiomatic characterization of a fuzzy generalization of rough sets	Generalized fuzzy rough approximation operators based on fuzzy coverings
6	On generalized intuitionistic fuzzy rough approximation operators	**Generalized lower and upper approximations in a ring**	Rough approximation operators on two universes of discourse and their fuzzy extensions
7	On characterization of generalized interval-valued fuzzy rough sets on two universes of discourse	Rough approximation operators on two universes of discourse and their fuzzy extensions	An axiomatic characterization of a fuzzy generalization of rough sets
8	Generalized fuzzy rough sets	A novel approach to fuzzy rough sets based on a fuzzy covering	A novel approach to fuzzy rough sets based on a fuzzy covering
9	Rough approximation operators on two universes of discourse and their fuzzy extensions	On characterizations of (*I*, *T*)-fuzzy rough approximation operators	On fuzzy rings
10	**The product structure of fuzzy rough sets on a group and the rough T-fuzzy group**	On characterization of generalized interval-valued fuzzy rough sets on two universes of discourse	On characterizations of (*I*, *T*)-fuzzy rough approximation operators
AP	0.18	0.772	0.853

with the selected paper as for example the first document ranked by *abstract*. On the other hand, the LDA initialization based on keyword clustering cause the difference between *LM0* and *LM2e*. More specifically, this is due to the fact that, in such a case, the keywords, although different sometimes, are grouped under the same clusters. When this happens, the words occurring in the abstracts of those documents are assumed by the LDA initialization to have been generated by the same topic, increasing the probabilities related to that given topic in both models and reducing the differences between them (as long as the weight given to the topics in (21) is big enough).

References

[1] I. Bíró, J. Szabó, A.A. Benczúr, Latent Dirichlet Allocation in web spam filtering, in: Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web, 2008, pp. 29–32.
[2] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet Allocation, Journal of Machine Learning Research 3 (2003) 993–1022.
[3] D.M. Blei, J.D. McAuliffe, Supervised topic models, in: Proceedings of the 21st Annual Conference on Neural Information Processing Systems, 2007, pp. 121–128.
[4] T. Bogers, A. van den Bosch, Recommending scientific articles using CiteULike, in: Proceedings of the 2008 ACM Conference on Recommender Systems, 2008, pp. 287–290.
[5] L. Bolelli, S. Ertekin, D. Zhou, C.L. Giles, Finding topic trends in digital libraries, in: Proceedings of the Joint International Conference on Digital Libraries, 2009, pp. 69–72.
[6] C.L. Clarke, M. Kolla, G.V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, I. MacKinnon, Novelty and diversity in information retrieval evaluation, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008, pp. 659–666.
[7] H. Deng, I. King, M.R. Lyu, Enhancing expertise retrieval using community-aware strategies, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, 2009, pp. 1733–1736.
[8] M. Franke, A. Geyer-Schulz, A. Neumann, Recommender services in scientific digital libraries, in: Multimedia Services in Intelligent Environments, Studies in Computational Intelligence, vol. 120, 2008, pp. 377–417.
[9] B. Fuglede, F. Topsoe, Jensen–Shannon divergence and Hilbert space embedding, in: Proceedings of the IEEE International Symposium on Information Theory, 2004, pp. 31–36.
[10] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007, pp. 1606–1611.
[11] P.J. Garcés, J.A. Olivas, F.P. Romero, Concept-matching IR systems versus word-matching information retrieval systems: considering fuzzy interrelations for indexing web pages, Journal of the American Society for Information Science and Technology 57 (4) (2006) 564–576.
[12] L. Geng, H. Wang, X. Wang, L. Korba, Adapting LDA model to discover author-topic relations for email analysis, in: Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery, 2008, pp. 337–346.
[13] T.L. Griffiths, M. Steyvers, Finding scientific topics, Proceedings of the National Academy of Sciences 101 (Suppl. 1) (2004) 5228–5235.
[14] L. Hong, B.D. Davison, Empirical study of topic modeling in Twitter, in: Proceedings of the First Workshop on Social Media Analytics, 2010, pp. 80–88.
[15] G. Hurtado Martín, S. Schockaert, C. Cornelis, H. Naessens, Metadata impact on research paper similarity, in: Proceedings of the 14th European conference on Research and Advanced Technology for Digital Libraries, 2010, pp. 457–460.
[16] G. Hurtado Martín, S. Schockaert, C. Cornelis, H. Naessens, Finding similar research papers using language models, in: Proceedings of the 2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation, 2011, pp. 106–113.
[17] F. Jelinek, R.L. Mercer, Interpolated estimation of Markov source parameters from sparse data, in: Proceedings of the Workshop on Pattern Recognition in Practice, 1980, pp. 381–397.
[18] M. Karimzadehgan, R.W. White, M. Richardson, Enhancing expert finding using organizational hierarchies, in: Proceedings of the 31th European Conference on IR Research, 2009, pp. 177–188.

[19] S. Kataria, K. Kumar, R. Rastogi, P. Sen, S. Sengamedu, Entity disambiguation with hierarchical topic models, in: Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2011, pp. 1037–1045.
[20] R. Krestel, P. Fankhauser, W. Nejdl, Latent Dirichlet Allocation for tag recommendation, in: Proceedings of the 2009 ACM Conference on Recommender Systems, 2009, pp. 61–68.
[21] S. Kullback, R.A. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1) (1951) 79–86.
[22] M. Lease, E. Charniak, A Dirichlet-smoothed bigram model for retrieving spontaneous speech, in: Proceedings of the Cross-Language Evaluation Forum, 2007, pp. 687–694.
[23] S.P. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.
[24] S.M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S.K. Lam, A.M. Rashid, J.A. Konstan, J. Riedl, On the recommending of citations for research papers, in: Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work, 2002, pp. 116–125.
[25] D.M. Mimno, A. McCallum, Expertise modeling for matching papers with reviewers, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 500–509.
[26] D. Petkova, W.B. Croft, Hierarchical language models for expert finding in enterprise corpora, in: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, 2006, pp. 599–608.
[27] X.H. Phan, M.L. Nguyen, S. Horiguchi, Learning to classify short and sparse text & web with hidden topics from large-scale data collections, in: Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 91–100.
[28] J.M. Ponte, W.B. Croft, A language modeling approach to information retrieval, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998, pp. 275–281.
[29] C. Porcel, A. Tejeda-Lorente, M.A. Martínez, E. Herrera-Viedma, A hybrid recommender system for the selective dissemination of research resources in a technology transfer office, Information Sciences 184 (1) (2012) 1–19.
[30] X. Quan, G. Liu, Z. Lu, X. Ni, L. Wenyin, Short text similarity based on probabilistic topics, Knowledge and Information Systems 25 (2010) 473–491.
[31] D. Ramage, D. Hall, R. Nallapati, C.D. Manning, Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009, pp. 248–256.
[32] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing and Management 24 (1988) 513–523.
[33] N.R. Smalheiser, V.I. Torvik, Author name disambiguation, Annual Review of Information Science and Technology 43 (2009) 1–43.
[34] M. Steyvers, P. Smyth, M. Rosen-Zvi, T.L. Griffiths, Probabilistic author-topic models for information discovery, in: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 306–315.
[35] J. Weng, E.-P. Lim, J. Jiang, Q. He, Twitterrank: finding topic-sensitive influential twitterers, in: Proceedings of the Third International Conference on Web Search and Web Data Mining, 2010, pp. 261–270.
[36] C. Zhai, J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001, pp. 334–342.
[37] D. Zhou, J. Bian, S. Zheng, H. Zha, C.L. Giles, Exploring social annotations for information retrieval, in: Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 715–724.
[38] J. Zhu, X. Huang, D. Song, S. Rüger, Integrating multiple document features in language models for expert finding, Knowledge and Information Systems 23 (2010) 29–54.