Ghent University
Faculty of Sciences
Department of Applied Mathematics, Computer Science and Statistics

Intelligent Methods for Information Filtering of Research Resources

Dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Computer Science

Germán Hurtado Martín

July 2013

Supervisors: Prof. dr. Chris Cornelis
Dr. Steven Schockaert
Prof. dr. Helga Naessens

Contents

Abstract

This thesis presents several content-based methods to address the task of filtering research resources. The explosive growth of the Web in the last decades has led to an important increase in available scientific information. This has contributed to the need for tools which help researchers to deal with huge amounts of data. Examples of such tools are digital libraries, dedicated search engines, and personalized information filters. The latter, also known as recommenders, have proved useful for non-academic purposes and in the last years have started to be considered for recommendation of scholarly resources. This thesis explores new developments in this context.

In particular, we focus on two different tasks. First we explore how to make maximal use of the semi-structured information typically available for research papers, such as keywords, authors, or journal, to assess research paper similarity. This is important since in many cases the full text of the articles is not available and the information used for tasks such as article recommendation is often limited to the abstracts. To exploit all the available information, we propose several methods based on both the vector space model and language modeling. In the first case, we study how the popular combination of tf-idf and cosine similarity can be used not only with the abstract, but also with the keywords and the authors. We also combine the abstract and these extra features by using Explicit Semantic Analysis. In the second case, we estimate separate language models based on each of the features to subsequently interpolate them. Moreover, we employ Latent Dirichlet Allocation (LDA) to discover latent topics which can enrich the models, and we explore how to use the keywords and the authors to improve the performance of the standard LDA algorithm.

Next, we study the information available in call for papers (CFPs) of conferences to exploit it in content-based methods to match users with CFPs. Specifically, we distinguish between textual content such as the introductory text and topics in the scope of the conference, and names of the program committee. This second type of information can be used to retrieve the research papers written by these people, which provides the system with new data about the conference. Moreover, the research papers written by the users are employed to represent their interests. Again, we explore methods based on both the vector space model and language modeling to combine the different types of information.

The experimental results indicate that the use of these extra features can lead to significant improvements. In particular, our methods based on interpolation of language models perform well for the task of assessing the similarity between research papers. On the contrary, when addressing the problem of filtering CFPs the methods based on the vector space model are shown to be more robust.

Samenvatting

Dit proefschrift stelt verschillende content-gebaseerde methoden voor om het probleem van het filteren van onderzoeksgerelateerde resources aan te pakken. De explosieve groei van het internet in de laatste decennia heeft geleid tot een belangrijke toename van de beschikbare wetenschappelijke informatie. Dit heeft bijgedragen aan de behoefte aan tools die onderzoekers helpen om om te gaan met grote hoeveelheden van data. Voorbeelden van dergelijke tools zijn digitale bibliotheken, specifieke zoekmachines, en gepersonaliseerde informatiefilters. Deze laatste, ook gekend als aanbevelingssystemen, hebben ruimschoots hun nut bewezen voor niet-academische doeleinden, en in de laatste jaren is men ze ook beginnen inzetten voor de aanbeveling van wetenschappelijke resources. Dit proefschrift exploreert nieuwe ontwikkelingen in deze context.

In het bijzonder richten we ons op twee verschillende taken. Eerst onderzoeken we hoe we maximaal gebruik kunnen maken van de semi-gestructureerde informatie die doorgaans beschikbaar is voor wetenschappelijke artikels, zoals trefwoorden, auteurs, of tijdschrift, om de gelijkenis tussen wetenschappelijke artikels te beoordelen. Dit is belangrijk omdat in veel gevallen de volledige tekst van de artikelen niet beschikbaar is en de informatie gebruikt voor taken zoals aanbeveling van artikels vaak beperkt is tot de abstracts. Om alle beschikbare informatie te benutten, stellen we een aantal methoden voor op basis van zowel het vector space model en language models. In het eerste geval bestuderen we hoe de populaire combinatie van tf-idf en cosinussimilariteit gebruikt kan worden met niet alleen de abstract, maar ook met de trefwoorden en de auteurs. We combineren ook de abstract met deze extra informatie door het gebruik van Explicit Semantic Analysis. In het tweede geval schatten we afzonderlijke taalmodellen die gebaseerd zijn op de verschillende soorten informatie om ze daarna te interpoleren. Bovendien maken we gebruik van Latent Dirichlet Allocation (LDA) om latente onderwerpen te ontdekken die de modellen kunnen verrijken, en we onderzoeken hoe de trefwoorden en de auteurs gebruikt kunnen worden om de prestaties van de standaard LDA algoritme te verbeteren.

Vervolgens bestuderen we de informatie beschikbaar in de call for papers (CFPs) van conferenties om deze te exploiteren in content-gebaseerde methoden om gebruikers te matchen met CFPs. Met name maken we onderscheid tussen tekstuele inhoud, zoals de inleidende tekst en onderwerpen in het kader van de conferentie, en de namen van de programmacommissie. Dit tweede type informatie kan gebruikt worden om de artikels geschreven door deze mensen te achterhalen, wat het systeem voorziet van bijkomende gegevens over de conferentie. Bovendien worden de artikels geschreven door de gebruikers gebruikt om hun interesses te voorstellen. Opnieuw onderzoeken we methoden gebaseerd op zowel het vector space model als op language models om de verschillende soorten informatie te combineren.

De experimentele resultaten tonen aan dat het gebruik van deze extra informatie kan leiden tot significante verbeteringen. In het bijzonder presteren onze methoden op basis van interpolatie van taalmodellen goed voor de taak van het beoordelen van de gelijkenis tussen wetenschappelijke artikels. Daarentegen zijn de methoden gebaseerd op het vector space model meer robuust voor het probleem van het filteren van CFPs.

Acknowledgments

In the summer of 2006, after working for about a year in the industry, I decided that the world of IT consultancy (at least as it seemed to be understood in Spain at the time) was not for me, and that I wanted to work in research. I started to look for such a job, and after searching for a while I decided to apply for a position in Belgium that sounded really interesting. Seven years later, and as a result of that decision, I finished writing this PhD thesis. The cover of this book shows my name; however, this work would not have been possible without the help and support of many people during this time. I want to thank them from these pages.

First and foremost I would like to thank my promotor Chris Cornelis. He was the first person who contacted me when I applied for the position, and since then he has guided me. I still recall the first emails that we exchanged in that July of 2006, the interview we had via Skype, or the visit to Ghent for an interview with the people from the University College Ghent (HoGent), where I would work. During this visit, he did not only host me at his place, but he also spent a lot of time with me preparing the interview. And after I got the position, he supported me during the several months that passed until I could solve some infernal paperwork regarding the recognition of my qualifications (the research and dealing with bureaucracy during those months might be a subject for a different PhD on its own). Seeing his degree of involvement even before we could actually start working together made me feel really supported and secure, but also made me be tremendously afraid of not meeting his expectations and disappointing him in any way. I deeply hope I have not. His input throughout these years has been essential and I owe him most things I know about doing research. Thank you very much, Chris.

I also want to thank my other promotor Steven Schockaert for introducing me to many techniques in the information retrieval domain and for the innumerable amount of good ideas provided. I have to admit that more than once (and twice) I felt like strangling him when, after carefully designing an implementation for one idea, he came up with something completely different that would make me reimplement several parts of the system, feelings that were even stronger when the sequence of events was something like "let's try A", "let's try B", and then "let's try this variation on A that we

discarded before". However, his ideas have been key for this work and his high standards have ensured the quality in our publications. He is with no doubt one of the most brilliant people I have ever met and it has been a true privilege to work with him.

During these years I have spent most of the time working at the Computer Science team of the Department of Applied Engineering Sciences of the HoGent (although both team and department are now part of the Ghent University, UGent) and I cannot forget them here.

First of all I would like to thank my promotor within the HoGent, Helga Naessens, for her support. Since the interview we had in September of 2006 she has always been enthusiastic about my research and probably I would have never got to work at the HoGent (and therefore to complete my PhD) if it had not been for her enthusiasm and trust in me. The day I arrived in Ghent to work and stay here for a long time was not an easy day, not only because of the long travel but also because of relocating to a new and different country. That day she invited me to her place with her family for a dinner (in which I proved to be a bit picky with certain carefully chosen and prepared meals, sorry once again Helga!), and she then drove me back home. This gave me a feeling of support and comfort which I have kept noticing during all these years, and which has been essential to be able to fully focus on my work and my research without any other kind of worries.

The above is also applicable to the head of the team within the HoGent, Geert Van hoogenbemt. That day of my arrival in Ghent he was the one who came to pick me up at the station and driving me to my new home (even if it was a 10-minute walk) and then taking me to Helga's dinner. Since then he has been always available for me for any kind of issue, from work-related paperwork to less usual stuff regarding my new, Belgian life, like for example explaining how the health system works or how I could fill in my annual tax declaration. Not to mention, of course, his trust in me when I applied for the position. I want to thank him specially for his patience and support during the aforementioned months previous to my arrival, in which I had to solve all those issues regarding my qualifications. I have to say that he has set the bar really high for any other boss that I will have in the future.

Of course I also would like to thank the rest of colleagues at the HoGent. Before my arrival I was a bit afraid that people would be a bit colder than Spain, but fortunately they proved me wrong and from the very first day they made me feel just like one of them. It has not only been a pleasure to work with them during these years, but I owe them all I know now about teaching.

I would not like to forget the people of the Computational Web Intelligence team at the UGent. Special thanks to Martine De Cock, head of the team and one of the first people I met in this adventure. Also, I would like to thank Etienne Kerre, head of the bigger Fuzziness and Uncertainty Modelling research unit, who was my promotor during the first years of my

PhD. And I would also like to include here the people who, along with Chris, Steven and Martine, participated in my experiments: Glad Deschrijver, Kim Bauters, Nele Verbiest, Olivier Van Laere, Gustavo Torres, Jeroen Janssen and Mike Nachtegael from the UGent; Chris Jones and Vlad Tanasescu from Cardiff University; Enrique Herrera-Viedma, Francisco Herrera and Joaquín Derrac from the University of Granada; Luis Martínez from the University of Jaén; and Dirk Vermeir from the Free University of Brussels (VIB).

I also want to thank the members of my committee for finding time to read my thesis and attending both my private and public defenses: Martine De Cock, Véronique Hoste, Enrique Herrera-Viedma, Bart Dhoedt, and Kris Coolsaet. I would like to thank Martine and Bart especially for their thorough reading of the thesis, ready to find even the smallest typo (any errors remaining are my own), Véronique for pointing me out several interesting language processing techniques that I will certainly consider in the future, Enrique for allowing me see the proposed methods as part of a bigger, real system, as well as for finding a moment in his busy schedule every time I have been in Granada, and Kris for the smooth organization of the PhD examination procedure and for his help with the administrative aspects.

Finally I wish to thank the closest people in my life. First, thanks to my friends both in Spain and in Belgium. Without you there to share beers, parties, laughs, or holidays, everything would have been much harder. And last but not least, thanks to my family, and most especially to my parents. They have always supported and believed in me, and without them I would not be who I am. Sorry that you have had to miss me during all this time. This thesis is for you.

Chapter 1

Introduction

The growing popularity of the World Wide Web has resulted in huge amounts of information and many new applications. Academia has not remained unconnected to this, and apart from making scientific information more accessible, a variety of new tools have emerged to help researchers in different ways. On the one hand, direct collaboration among researchers has benefited from general purpose tools, such as emails or videoconferencing, or other ones designed for a specific task but not limited to academia, like project management applications, version control systems (e.g. CVS[1], Subversion[2]), or online editors (e.g. Google Docs[3], ShareLaTeX[4]) which make it easier for researchers to work on a specific project or article at the same time. On the other hand, other tools allow researchers to collaborate indirectly, by sharing their knowledge with others. As an example of this, bookmarking sites in particular have become very popular. On these sites, researchers can bookmark those sites or articles which they find interesting and then share them with other people, either explicitly by sending a link or implicitly by being followed by other researchers. Examples of such sites are BibSonomy[5], which allows the user to bookmark interesting websites and publications, or CiteULike[6], to manage and search scholarly references, encouraging researchers to discover new ones thanks to its recommendation service.

Another valuable resource for research are digital libraries, which manage and help dealing with the vast amounts of scientific literature generated each year. Some publishers make their publications available online so they can be consulted from anywhere at any moment (although some content is only viewable for subscribers); examples of such libraries are the ACM Digital

[1]http://savannah.nongnu.org/projects/cvs
[2]http://subversion.apache.org
[3]http://docs.google.com
[4]http://www.sharelatex.com
[5]http://www.bibsonomy.org
[6]http://www.citeulike.org

Library[7] or Elsevier's ScienceDirect[8]. These sites usually include a search engine. Additionally, the content of many of these libraries is also indexed by external, specialized search engines such as Google Scholar[9] or Microsoft Academic Search[10], which index the repositories of academic institutions too. Publications by researchers from the whole world are then just a couple of clicks away.

However, the power of these search engines is not always enough to find the desired information, since users have to express their information needs by means by a query consisting of only a few terms, which cannot always capture all the aspects of what they are really looking for. While advances in information retrieval try to improve the techniques behind these systems, another alternative is to rely on information filtering instead. In this case, the system filters out all information not considered as relevant to the users, so they are presented only with potentially interesting information. For example, in the case of research papers, users are only shown those papers which might be relevant for their research. User interests are usually represented by a profile, or are inferred from his actions (e.g. if the user is browsing a given paper, the system can present similar papers since possibly he is interested in that topic). Recommender systems are inspired by this idea.

In the last years, recommendation of research resources has gained popularity, especially research paper recommendation, and several methods to address such tasks have been studied. However, it is a relatively new research domain; commercial systems have only recently started to use such techniques to offer recommendation services, and there are still many possibilities which remain unexplored.

The goal of this thesis is to study new methods that can be applied to filter research resources. On the one hand, we present several content-based methods to assess research paper similarity. These methods can be employed, for instance, for research paper recommendation. In this case, the methods can be used to find papers similar to a given paper interesting to the user. Also, if each user is profiled by means of his own research papers (i.e., we assume that the research papers that he has written represent his interests), such methods can be used to compare other papers to those in the set and therefore offer personal recommendations. On the other hand, we aim for a specific task: filtering calls for papers (CFPs) of scientific conferences. This is a problem which to our knowledge has not been addressed yet, but which could be an attractive addition to the tools available to researchers. Such an ideal system for filtering CFPs would use several methods to deal with different types of information; in this thesis

[7]http://dl.acm.org
[8]http://www.sciencedirect.com
[9]http://scholar.google.be
[10]http://academic.research.microsoft.com

we study the content-based methods only. In particular, we examine how different features of a typical CFP can be modeled and compared, for which we also use, as part of the techniques, the methods for assessing research paper similarity previously studied.

Specifically, we focus on the different kinds of information that can be found in a document and how they can be used to improve the assessment of document similarity. In the case of research papers this is particularly important since the full text is often not available, and the publicly available content is then limited to the abstract and other features such as keywords, authors, or journal. It is therefore desirable to make optimal use of them, and we propose several methods to exploit that information. Moreover, these features do not only add useful information to the document or user representations, but can also be used to access extra information. For example, in the case of research papers, keywords and author names can be used to help discovering latent information. On the other hand, in the case of CFPs, the names of the members of the program committee can be used to retrieve the papers that they have written, and these in turn can be used to enrich the representation of a CFP in the system.

1.1 Thesis outline

This thesis is structured as follows. Chapter 2 introduces basic ideas from information retrieval, focusing especially on the methods used as our basis in Chapters 4 and 5, but which are also necessary to fully understand other approaches reviewed in Chapter 3. In particular, we describe the vector space model, in which documents are represented as vectors, and language models, based on probabilistic models. In both cases we discuss several methods within those frameworks, and we pay special attention to how they can be applied to assessing document similarity.

In Chapter 3 we take a look at the information filtering domain, introducing and reviewing some basic concepts, to subsequently focus on information filtering of research resources. Specifically, we present a broad survey of the research carried out on this domain in the last years, with emphasis on the recommendation of research papers. Finally, we see how some of this research has been applied to actual systems used by thousands of people every day.

Chapter 4 proposes novel methods to assess research paper similarity. In particular, we focus on content-based approaches that exploit a number of features usually available for research papers such as keywords, authors, or journal. Some of these methods are based on the vector space model; more specifically, we follow a well-known approach that we use as our baseline, in addition to another model based on Explicit Semantic Analysis. On the other hand, we also explore how language modeling can be used to combine the information from the various features. Also, we use Latent Dirichlet Allocation to discover latent topics, and propose several methods to enhance this technique.

In Chapter 5 we study content-based methods for filtering calls for papers (CFPs) of conferences. To our knowledge, filtering this kind of resources was previously unexplored, and it allows to explore how to apply some of the ideas from the previous chapter to a specific task. As for research papers, in CFPs we still find different features such as an introductory text about the conference or the names of the people in the program committee. We also examine how we can take advantage of information about the research papers that have been written by members of the program committee or by users of the system, in order to better characterize the scope of a conference or the interests of a user.

Finally, in Chapter 6 we summarize the conclusions of this thesis and present some possible directions for future research.

We lastly note that part of the research results published in this thesis have been presented in international journals [71] and in the proceedings of international conferences [65, 66, 67, 68, 69, 70].

Chapter 2

Preliminaries from Information Retrieval

In this chapter we introduce some basic ideas related to information retrieval which are used in this thesis. We particularly focus on those specific methods used in our work. Specifically, we first recall the vector space model, a model that, as reflected by its name, represents documents as vectors. Also, we recall two different approaches to calculate the components of those vectors, namely tf-idf and Explicit Semantic Analysis (ESA). Then we discuss language modeling, an alternative to the vector space model based on probabilistic models. Finally we describe Latent Dirichlet Allocation (LDA), another probabilistic method that attempts to discover the latent topic structure in a document collection. For the sake of completeness, we end the chapter with a brief review of other general methods for assessing text document similarity.

2.1 Vector space model

The vector space model [124] is an algebraic model that represents text documents as vectors. A document d is then represented as vector $d = (w_1, w_2, \cdots, w_n)$, where each component w_i contains a weight corresponding to each of the different terms occurring in d. Each weight reflects the importance of that term in the document and/or in a given collection of documents. The similarity between two documents can then be assessed simply by comparing their vectors. The whole procedure can therefore be divided in three steps: establishing the terms that determine the components of the vectors, computing weights for these terms, and comparing the resulting vectors by means of a given similarity measure.

2.1.1 Defining the terms

The definition of *term* can vary, depending on the pursued goal. We can consider each single word to be a term, in this case we talk about *unigrams.* Another option might be to consider keywords, that can contain more than one word each (e.g. "information retrieval"). This is an example of *multigrams.* In general, a multigram is any term containing more than one word, so it can also be a name, an expression, a phrase, or a whole sentence. More specific kinds of multigrams are fixed-length multigrams, like *bigrams* (containing two words), *trigrams* (three words), and *n-grams* in general, where n is the number of single words contained in the term.

For example, let document d be $d = \{$*recommender systems and intelligent systems in general for paper recommendation: building a research paper recommender focused on artificial intelligence*$\}$. If we decide to work with unigrams, the vector representation of d will have sixteen components, one for each of the different words (there is a total of nineteen words but *recommender*, *systems* and *paper* occur twice). However, if we do not want to truncate the phrases (e.g. we want *recommender systems* to be a term), we can work with multigrams. A typical option is to consider all sequences of n words in the text, although this also leads to terms that are not real phrases (e.g. *systems and*, for $n = 2$). A more elaborated approach is to work with a specific vocabulary, and for a given range of values of n consider only those n-grams that refer to a term listed in it. In this case, the vector representation of d contains thirteen components: *recommender systems*, *and*, *intelligent systems*, *in*, *general*, *for*, *paper recommendation*, *building*, *a*, *research paper recommender*, *focused*, *on*, and *artificial intelligence.*

2.1.1.1 Stopword removal

We can see that in the vector representation of d some components are assigned to short function words, i.e., words with an important grammatical function but which outside a sentence are not really useful, like *a*, *and*, or *for.* These words are called *stopwords* and can be safely removed without altering the quality of the information in the document. Forms of common verbs (e.g. *to be*) are usually considered stopwords too and therefore removed as well. This results in shorter vectors, which speeds up computations and in some cases also leads to better results since "noise" has been removed. Since there is no unique, standard list of stopwords, this needs to be defined for each application[1]. In this case, we consider articles, conjunctions and prepositions to be stopwords. The number of components in the resulting vector drops from sixteen to eleven.

[1]There are, however, some lists which are commonly used, or that can serve as a basis for a new list by removing or adding terms. An example of this can be found at http://snowball.tartarus.org/algorithms/english/stop.txt

2.1.1.2 Stemming

One of the limitations of the vector space model is the fact that a sufficient number of terms of two document vectors d_1 and d_2 must be identical for the similarity between d_1 and d_2 to be considered high. However, two related documents often contain similar, but not identical terms. In our example, this is illustrated by *recommender* and *recommendation*, as well as by *intelligent* and *intelligence*. Something similar happens with verbs in different forms and tenses. To overcome this problem, words can be reduced to their root or stem; this technique is known as *stemming* [98]. Hence, if we apply stemming to our example using unigrams and stopword filtering, we get a vector with components for the following terms: *recommend*, *system*, *intelligen*, *general*, *paper*, *build*, *research*, *focus*, and *artificial*. Note how the plural forms disappear, and how the verbs also change. The resulting terms need not be actual words, like *intelligen*.

2.1.1.3 Feature selection

While stopword removal and stemming can help reduce the number of terms in the documents, leading to shorter vectors, sometimes the dimensionality of the feature space (i.e., the number of different terms in the collection) is still too high, which means a less efficient but also a less robust system. Therefore, more complex techniques must be used to tackle this problem. These techniques can be grouped under the concept of feature selection.

Feature selection is a process that chooses a subset from the original feature set according to some criteria [94]. The idea is that the selected subset still retains most of the information contained in the original set. In other words, what the process does is to identify and remove those terms that do not contain a significant amount of information. This enhances efficiency and robustness without a negative impact on the final results.

There are different feature selection techniques; in this section we only discuss the *term strength* method, as it will be used in subsequent chapters. We have chosen this method because it is unsupervised and because most other methods are intended for classification, where documents are subdivided in different classes. For a complete study on feature selection methods we refer to [149].

The *term strength* method is based on the idea that terms shared by closely related documents are more informative than others [149]. The strength of a term w is thus computed by estimating the probability that a term w occurs in a document d_1 given that it occurs in a related document d_2:

$$strength(w) = P(w \in d_1 | w \in d_2) \tag{2.1}$$

This probability can be estimated by the percentage of pairs of related documents (d_1, d_2) where w occurs in both documents.

Ideally, the pairs of related documents are already known, for example because they have previously been annotated by experts. However, in many cases these pairs are unknown; the first step is then to choose pairs of related documents, which can be done by using some approach to compute the similarity between two documents (for example, any of the approaches explained in the next sections). In this case, to define how close two documents must be to be considered as related, a threshold is used, namely the average number of related documents per document. This means that a similarity score is set as a minimum for considering two documents as related, and all documents are compared using the chosen approach. If the average number of related documents per document is above the threshold (i.e., too many related documents per document), the minimum similarity score is raised, and the process is repeated until the average number of related documents is below the threshold. Since a too small number of related documents is not desirable either, a second threshold can be used to prevent that. According to [149], satisfactory performance is achieved when using a threshold between 10 and 20.

When the pairs of related documents are known, and after calculating $strength(w)$ for every term w in the document collection, the N strongest terms are selected, ignoring the rest.

2.1.2 Computing the weights

As mentioned before, a document vector does not contain the terms themselves, but a weight corresponding to that term to reflect the importance of the term in the document and/or in a given collection of documents: the higher the weight, the more important the term is and the better it represents the document.

According to [122], there are three main factors to take into account when computing the weights: term frequency factor, collection frequency factor, and length normalization factor.

On the one hand, it seems obvious that the most frequently mentioned terms in a document are important, therefore the term frequency (*tf*) is an interesting metric for computing the weight. The term frequency of term w_i in document d can be calculated as:

$$tf(w_i, d) = \frac{n(w_i, d)}{|d|} \tag{2.2}$$

where $n(w_i, d)$ is the number of occurrences of w_i in d and $|d|$ is the total number of terms in d.

On the other hand, if a term appears in most documents, it cannot be seen as a discriminative term, regardless of its actual importance. In these cases the collection frequency factor works better: by calculating the inverse document frequency (*idf*) we can get higher weights for those terms that appear only in a few documents:

$$idf(w_i, d) = \frac{|\mathcal{C}|}{|\{d_j \in \mathcal{C} : w_i \in d_j\}|} \tag{2.3}$$

where $|\mathcal{C}|$ is the number of documents in the collection, and $|\{d_j \in \mathcal{C} : w_i \in d_j\}|$ is the number of documents in the collection that contain w_i.

The strengths of both metrics can be combined in the tf-idf weight:

$$tfidf(w_i, d) = \frac{n(w_i, d)}{|d|} \cdot \log(\frac{|\mathcal{C}|}{|\{d_j \in \mathcal{C} : w_i \in d_j\}|}) \tag{2.4}$$

The logarithm is introduced to smooth the influence of the idf value; a term occurring in 10 times more documents than another should indeed lead to a lower tf-idf value, but a value 10 times smaller is too drastic. This can be avoided using the logarithm: tf-idf values are still proportional to idf but in a less harsh way. A common alternative to (2.4), used to avoid divisions by zero in thoses cases when w_i is not in the collection, is:

$$tfidf(w_i, d) = \frac{n(w_i, d)}{|d|} \cdot \log(\frac{|\mathcal{C}|}{|\{d_j \in \mathcal{C} : w_i \in d_j\}| + 1}) \tag{2.5}$$

Finally, the length normalization factor must be considered, since not all documents are equally long and this may lead to unfair comparisons. Therefore, after computing each vector $\mathbf{d}$, it should be normalized: $\mathbf{d} = (\frac{w_1}{\|\mathbf{d}\|}, \cdots, \frac{w_n}{\|\mathbf{d}\|})$, where $\|\mathbf{d}\|$ is the Euclidean norm $\|\mathbf{d}\| = \sqrt{{w_1}^2 + \cdots + {w_n}^2}$.

2.1.3 An alternative: Explicit Semantic Analysis

The tf-idf weighting scheme is the most popular approach in information retrieval, due to its good performance and simplicity. However, only lexical similarity is taken into account. In this section we therefore focus on Explicit Semantic Analysis (ESA) [46], an approach that does not only deal then with lexical information, but includes semantic information too. Instead of words, the components in the vectors used by ESA refer to concepts: a document is represented not as a weighted vector of words, but as a weighted vector of concepts. However, to do so the concepts must be previously defined, which means that an extra source of information other than the modelled documents is required. In particular, [46] proposes to use Wikipedia[2] to define the concepts.

More formally, in this scheme, a vector representation $\mathbf{d_E}$ is defined for each document d, where $\mathbf{d_E}$ has one component for every concept c in Wikipedia. The idea is that each component of the vector should reflect how related the document is to the corresponding concept.

[2]http://en.wikipedia.org

Let $\mathbf{d}$ be the vector obtained for document d by using the tf-idf scheme. In addition, we consider a vector $\mathbf{d_c}$ to represent each concept c. In order to build such a vector, the collection $\mathcal{C}_E$ of Wikipedia pages is considered; this collection contains a document d_c for each concept. In the weighted vector $\mathbf{d_c}$ each component corresponds to a term in $\mathcal{C}_E$ (i.e., a term occurring in at least one Wikipedia page), and the weights are the tf-idf scores calculated w.r.t. $\mathcal{C}_E$. Thus, $\mathbf{d_c}$ represents Wikipedia concept c in the same way that $\mathbf{d}$ represents document d. Finally, $\mathbf{d}$ and $\mathbf{d_c}$ are normalized and can be compared to compute the new vector representation $\mathbf{d_E}$ of document d. In particular, the weight w_c in $\mathbf{d_E}$ of the component corresponding to concept c is calculated as follows:

$$w_c = \mathbf{d} \cdot \mathbf{d_c} \tag{2.6}$$

where $\mathbf{d} \cdot \mathbf{d_c}$ denotes the scalar product. The whole process is summarized in Figure 2.1.

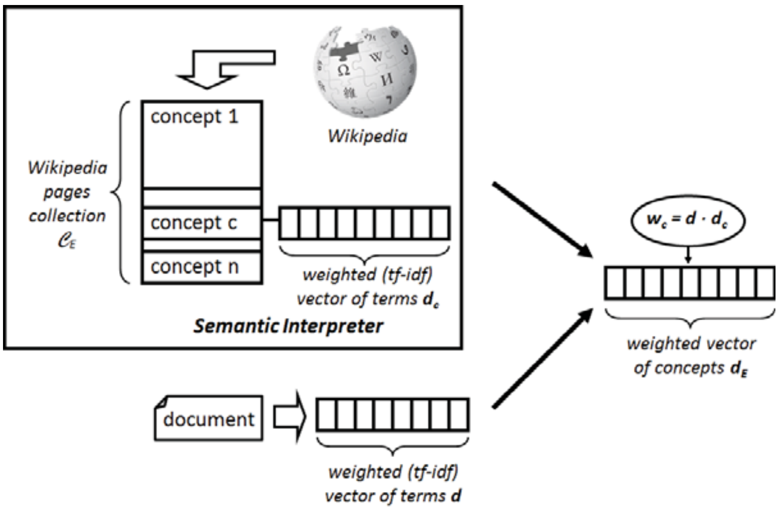

concept 1
Wikipedia
Wikipedia pages collection $\mathcal{C}_E$
concept c
concept n
weighted (tf-idf) vector of terms d_c
Semantic Interpreter
$w_c = d \cdot d_c$
weighted vector of concepts d_E
document
weighted (tf-idf) vector of terms d

Figure 2.1: Wikipedia-based generation of the ESA vector $\mathbf{d_E}$ of a document

2.1.4 Comparing the vectors

Once the weighted vectors have been constructed, their similarity can be calculated. Two vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ corresponding to different papers can be compared using standard similarity measures. The most commonly used similarity measure is the cosine similarity, defined by:

$$sim_c(\mathbf{d_1}, \mathbf{d_2}) = \frac{\mathbf{d_1} \cdot \mathbf{d_2}}{\|\mathbf{d_1}\| \cdot \|\mathbf{d_2}\|} \tag{2.7}$$

where, again, $\mathbf{d_1} \cdot \mathbf{d_2}$ denotes the scalar product and $\|.\|$ is the Euclidean norm. The cosine similarity measures the angle between $\mathbf{d_1}$ and $\mathbf{d_2}$: the larger the angle between the vectors, the less similar the documents that they represent. To this end, (2.7) is derived from the formula of the scalar product:

$$\mathbf{d_1} \cdot \mathbf{d_2} = \|\mathbf{d_1}\| \cdot \|\mathbf{d_2}\| \cdot \cos\theta \tag{2.8}$$

The cosine similarity is then, as its name indicates, equal to $\cos\theta$. Since the weights of the components of the vectors cannot be negative, the result is always a number between 0 (vectors form a 90° angle, i.e., they are completely different) and 1 (the vectors are identical).

Another measure based on the same idea is the Dice similarity [39], defined by

$$sim_d(\mathbf{d_1}, \mathbf{d_2}) = \frac{2(\mathbf{d_1} \cdot \mathbf{d_2})}{\|\mathbf{d_1}\|^2 \cdot \|\mathbf{d_2}\|^2} \tag{2.9}$$

Note that the denominator in both (2.7) and (2.9) as well as the norms in the right-hand side of (2.8) are unnecessary when $\mathbf{d_1}$ and $\mathbf{d_2}$ are normalized.

Finally, two well-known alternatives which focus more directly on the overlap of the two vectors are those based on the Jaccard index [72]. The original Jaccard index compares two sets by dividing the size of the intersection of the two sets by the union of the two sets. This idea can be applied to compare weighted vectors in two different ways. On the one hand, the generalized Jaccard similarity [54], defined by

$$sim_{gj}(\mathbf{d_1}, \mathbf{d_2}) = \frac{\sum_k \min(d_{1_k}, d_{2_k})}{\sum_k \max(d_{1_k}, d_{2_k})} \tag{2.10}$$

straightforwardly adapts the original idea: it compares the sum of the weights shared by the two vectors (e.g. if $d_{1_k} = 0.3$ and $d_{2_k} = 0.1$ the shared weight for that term is 0.1) to the sum of the weights obtained when both vectors are considered (e.g. 0.3 for the same case of $d_{1_k} = 0.3$ and $d_{2_k} = 0.1$). On the other hand, the extended Jaccard similarity [121], defined by

$$sim_{ej}(\mathbf{d_1}, \mathbf{d_2}) = \frac{\mathbf{d_1} \cdot \mathbf{d_2}}{\|\mathbf{d_1}\|^2 + \|\mathbf{d_2}\|^2 - (\mathbf{d_1} \cdot \mathbf{d_2})} \tag{2.11}$$

compares the total sum of the weights of the terms shared by both vectors to the sum of the weights of the terms that only occur in one of the vectors.

2.2 Language modeling

Weighted vectors are not the only way to represent text documents. Among other approaches to this end, language modeling has received much attention in the last years, and has been shown to perform well for comparing short text snippets [62, 118], which as we will see in the following chapters makes it interesting for our purposes.

Language modeling is based on estimating the probability distribution of a document, where, as in the previous section, a term can be either a unigram or a multigram. This estimated probability distribution represents the language model of the document. Since language modeling is used in a broad range of applications, the definition of a term, as well as the ways to estimate the language model vary widely depending on the pursued goal. For example, in natural language processing applications it is common to consider multigrams, since a given group of words can help to predict the next word in a sentence. In other cases, such as document classification, models are not only estimated for the documents, but also for the classes. Since this thesis is about information retrieval, in the remainder of this section we focus exclusively on the approach for this kind of task.

In the language modeling approach to information retrieval [117], a unigram language model D is estimated for each document d in a collection $\mathcal{C}$. The idea, given a query q, is first to calculate, for each document d with a model D, the probability $P(q|D)$ that the language model D could generate the terms in q. After that, a list of documents is retrieved: the more likely a model is to have generated the query, the higher ranked the corresponding document is in the list. In other words, we are assuming that if model D (ideally) generated document d, the higher the probability of it having generated query q, the more related d and q are.

In order to estimate the language model D for a document d, we therefore have to estimate the probability $P(w|D)$ for each term w in d. The maximum likelihood estimate of this probability is:

$$P(w|D) = \frac{n(w,d)}{|d|} \tag{2.12}$$

where $n(w,d)$ is the number of occurrences of w in d, and $|d|$ is the total number of terms in d. As in the vector space model, it is possible to first filter the terms, removing stopwords and/or applying stemming.

To illustrate how language modeling works, we take up the document d used in Section 2.1.1, $d = \{$*recommender systems and intelligent systems in general for paper recommendation: building a research paper recommender focused on artificial intelligence*$\}$. Assuming that we use unigrams and that we remove stopwords, the resulting set of terms is $d = \{$*recommender, systems, intelligent, systems, general, paper, recommendation, building, research, paper, recommender, focused, artificial, intelligence*$\}$

Now, given a query $q = \{$*recommender systems*$\}$, we can calculate the query likelihood $P(q|D)$. Since we are using unigrams and assuming that terms are independent, a simple method to calculate $P(q|D)$ is just to multiply the probabilities of each word in the query:

$$P(q|D) = \prod_{i=1} P(w_i|D) \tag{2.13}$$

In this case, we have $P(recommender|D) = \frac{2}{14}$ and $P(systems|D) = \frac{2}{14}$, so $P(q|D) = 0.02$. If we now change our query to $q = \{$*scientific recommender systems*$\}$, we add $P(scientific|D)$ to the product. However, as *scientific* is not in d, $P(scientific|D) = 0$ and $P(q|D) = 0$, which is unreasonable. The language model D should therefore consider terms that are not in d; this technique is called *smoothing.*

Since we are not dealing with just one document but with many documents in a collection $\mathcal{C}$, a common solution to smooth the models is to combine the document model $P(w|D)$ with the collection model $P(w|\mathcal{C})$, estimated as:

$$P(w|\mathcal{C}) = \frac{n(w, \mathcal{C})}{|\mathcal{C}|} \tag{2.14}$$

where $n(w, \mathcal{C})$ is the number of occurrences of w in the collection, and $|\mathcal{C}|$ is the total number of terms in the collection. Note the similarities with the tf-idf weighting scheme in the vector space model: $P(w|D)$ works with the term frequency, while $P(w|\mathcal{C})$ uses the collection frequency. A simple method is Jelinek-Mercer smoothing [152], which linearly interpolates both models:

$$P^*(w|D) = \lambda P(w|D) + (1 - \lambda)P(w|\mathcal{C}) \tag{2.15}$$

where parameter $\lambda \in [0, 1]$ controls the weight given to each model.

A common alternative to Jelinek-Mercer smoothing is Bayesian smoothing [152], also referred to as Bayesian smoothing with Dirichlet priors or simply Dirichlet smoothing. In this case, the model is built using the Dirichlet prior and model parameter μ:

$$P^*(w|D) = \frac{n(w, d) + \mu P(w|\mathcal{C})}{|d| + \mu} \tag{2.16}$$

As it can be noticed, unlike Jelinek-Mercer smoothing, this type of smoothing depends on the length of the document, which makes sense as intuitively longer documents contain more information and therefore the estimations require less smoothing. The value of μ is also related to the length of the documents, ranging from 0, which turns (2.16) into (2.12) (i.e., no smoothing), to a value several times $|d|$, which means that $P^*(w|D)$ is estimated almost solely based on the smoothing. The optimal value varies depending on the collection; a commonly used value is the average document length of the collection [115, 43].

For more information on smoothing and other smoothing methods we refer to [152] and [33].

As stated above and shown in the example, we can see how related a document d is to a query q according to (2.13) (where $P(w_i|D)$ can alternatively be replaced by $P^*(w_i|D)$ as defined in 2.15 or in 2.16). We can use this idea for documents instead of queries, making it possible to evaluate how similar two documents d_1 and d_2 are, by calculating $\prod_{i=1} P(w_i|D_1)$, $w_i \in d_2$, or $\prod_{i=1} P(w_i|D_2)$, $w_i \in d_1$.

Alternatively, we can compare the documents' models, D_1 and D_2. To do so, we can measure their difference using the Kullback-Leibler divergence [87], defined by

$$KLD(D_1||D_2) = \sum_w P^*(w|D_1) \log \frac{P^*(w|D_1)}{P^*(w|D_2)} \tag{2.17}$$

Intuitively, KLD measures the extra number of bits required for encoding data sampled from a distribution p using a code based on a second distribution q, which here could be seen as the extra information necessary to obtain a document originally generated by D_1 by using D_2 to generate it. 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 [45]. In this case, the models are first compared to an average model D_{avg} where the probability for each term w is estimated by

$$P^*(w|D_{avg}) = \frac{P^*(w|D_1) + P^*(w|D_2)}{2} \tag{2.18}$$

and then the mean of both divergences is calculated:

$$JSD(D_1||D_2) = \frac{KLD(D_1||D_{avg}) + KLD(D_2||D_{avg})}{2} \tag{2.19}$$

2.3 Latent Dirichlet Allocation

The standard language modeling approach does not measure semantic similarity, and therefore synonyms or related words are considered as totally different. An approach based on probabilistic models which deals with this problem is Latent Dirichlet Allocation (LDA) [21].

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.

Figure 2.2 shows an example of how this works. Document *doc* is assumed to be generated by the latent set of topics $T = \{T1, T2, T3\}$. Each of these topics has a different probability in the probability distribution θ, and its own probability distribution ϕ_i over (some of) the terms in the collection $\mathcal{C} = \{a, b, c, d, e, f\}$. To generate the first term of the document, a topic is sampled from θ, for example $T1$, and then a term is sampled from ϕ_1, c. Thus, the first term of the document is c. For the second term, a topic is

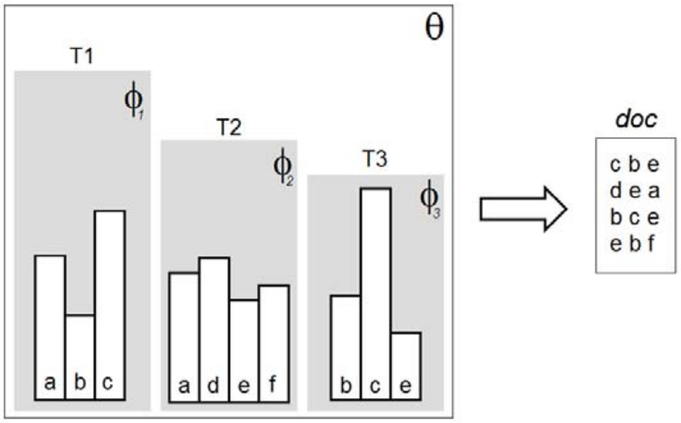


Figure 2.2: Generation of a document according to LDA

sampled from θ, $T3$, and a term is sampled from ϕ_3, b. The same process is repeated to generate each term in the document.

Therefore, if we want to represent a document according to the topics covered by it, the set of distributions ϕ over the terms in the collection (one distribution for each topic) and the set of distributions θ over all the topics (one distribution for each document) need to be estimated. To do so, we use LDA with Gibbs sampling [55]. These probabilities are then estimated as:

$$P(w|z) = \hat{\phi}_z^{(w)} = \frac{n_z{}^{(w)} + \beta}{n_z{}^{(\cdot)} + W\beta} \tag{2.20}$$

i.e., the probability that topic z generates term w, and:

$$P(z|\tau) = \hat{\theta}_z^{(d)} = \frac{n_z{}^{(d)} + \alpha}{n.^{(d)} + T\alpha} \tag{2.21}$$

the probability that topic z is sampled given τ, 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 (2.20) and (2.21) are to a maximum likelihood estimation: if their value is zero, (2.20) and (2.21) become a maximum likelihood estimation, while high values make them tend to a uniform distribution. Typical values for these parameters are $\alpha = 50/T$ and $\beta = 0.1$ as proposed in [55]. The number of topics T depends on the data and therefore differs for each problem. A typical and straightforward solution is simply trying different values to see which one offers the best results for the desired task. Alternatively, the likelihoods can be compared [55]. Finally, a third and more formal approach is to use Bayesian nonparametrics, specifically using hierarchical Dirichlet processes [137], although in practice this is less used due to its high computational cost.

The rest of the values in (2.20) and (2.21) are described in the first block of Table 2.1. All these values, except $n_{\cdot}^{(d)}$, which is simply the length of d, are unknown a priori.

Table 2.1: Values used in LDA with Gibbs sampling to find underlying topics

value	*description*
${n_z}^{(w)}$	Number of times term w is assumed to have been generated by topic z.
${n_z}^{(d)}$	Number of times a term instance of document d is assumed to have been generated by topic z.
${n_z}^{(\cdot)}$	Total number of times a term has supposedly been generated by topic z.
$n_{\cdot}^{(d)}$	Total number of term instances of document d generated by any topic.
${n'_z}^{(w)}$	Number of times term w is assumed to have been generated by topic z, but without counting the current assignment of w.
${n'_z}^{(d)}$	Number of times a term instance of document d is assumed to have been generated by topic z, but without counting the current assignment of w.
${n'_z}^{(\cdot)}$	Total number of times a term has supposedly been generated by topic z, but without counting the current assignment of w.
${n'_{\cdot}}^{(d)}$	Total number of term instances of document d generated by any topic, but without counting the current assignment of w.

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, 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

$$P(z|w,\tau) \propto P(w|z) \times P(z|\tau) = \frac{{n'_z}^{(w)} + \beta}{{n'_z}^{(\cdot)} + W\beta} \cdot \frac{{n'_z}^{(d)} + \alpha}{{n'_{\cdot}}^{(d)} + T\alpha} \tag{2.22}$$

Counts ${n'_z}^{(w)}$, ${n'_z}^{(d)}$, ${n'_z}^{(\cdot)}$ and ${n'_{\cdot}}^{(d)}$ are described in the second block of Table 2.1. When the algorithm stops after a specific number of iterations given as input, ϕ and θ can finally be estimated according to (2.20) and (2.21). Algorithm 1 shows the pseudo-code for the Gibbs sampling algorithm for LDA[3].

Now we can evaluate the probability $P(w|\tau)$ that term w is generated by the topics underlying document d:

$$P(w|\tau) = \sum_{i=1}^{T} P(w|z_i) \times P(z_i|\tau) \tag{2.23}$$

This allows us to reformulate (2.15) to build a model D not based on the text itself, but on the latent topics:

$$P^*(w|D) = \lambda P(w|\tau) + (1 - \lambda) P(w|\mathcal{C}) \tag{2.24}$$

As before, the documents' models can be compared by using (2.17) or (2.19).

[3]The pseudo-code of Algorithm 1 is based on both the description in [55] and Gregor Heinrich's Java code that can be found at http://arbylon.net/projects/LdaGibbsSampler.java

Algorithm 1 Gibbs sampling algorithm for LDA

```
1: ${n_z}^{(w)} = 0$ for all $z$, $w$    ▷ set all counts to zero
2: ${n_z}^{(d)} = 0$ for all $z$, $d$
3: ${n_z}^{(\cdot)} = 0$ for all $z$
4: for all document $d$ in $D$ do    ▷ initialize counts
5:     for all term ${w_i}^d$ in $d$ do
6:         sample $z$ from $\{z_1, \cdots, z_T\}$;    ▷ randomly sample a topic for ${w_i}^d$
7:         ${n_z}^{(w)} + +$;    ▷ topic $z$ has generated an instance of term $w$...
8:         ${n_z}^{(d)} + +$;    ▷ ... in document $d$
9:         ${n_z}^{(\cdot)} + +$;    ▷ increase number of terms generated by $z$
10:     end for
11:     $n.^{(d)} = |d|$;
12: end for    ▷ end of random initialization
13: for $num.iterations$ do
14:     for all document $d$ in $D$ do
15:         for all term ${w_i}^d$ in $d$ do
16:             remove instance ${w_i}^d$ from ${n_z}^{(w)}$    ▷ ${n_z}^{(w)}$ is now ${n'_z}^{(w)}$
17:             remove instance ${w_i}^d$ from ${n_z}^{(d)}$    ▷ ${n_z}^{(d)}$ is now ${n'_z}^{(d)}$
18:             remove instance ${w_i}^d$ from ${n_z}^{(\cdot)}$    ▷ ${n_z}^{(\cdot)}$ is now ${n'_z}^{(\cdot)}$
19:             $n.^{(d)} - -$;    ▷ $n.^{(d)}$ is now ${n'_\cdot}^{(d)}$
20:             sample $\hat{z}$ according to (2.22)
21:             ${n_{\hat{z}}}^{(w)} + +$;    ▷ topic $\hat{z}$ has now generated instance ${w_i}^d$
22:             ${n_{\hat{z}}}^{(d)} + +$;
23:             ${n_{\hat{z}}}^{(\cdot)} + +$;
24:             $n.^{(d)} + +$;
25:         end for
26:     end for    ▷ end of sampling process
27: end for    ▷ end of sampling
28: for all term $w$ in $\mathcal{C}$ do    ▷ distributions $\phi$ and $\theta$ are estimated
29:     for all topic $z$ in $\{z_1, \cdots, z_T\}$ do
30:         estimate $P(w|z)$ according to (2.20)
31:     end for
32: end for
33: for all document $d$ in $D$ do
34:     for all topic $z$ in $\{z_1, \cdots, z_T\}$ do
35:         estimate $P(z|\tau)$ according to (2.21)
36:     end for
37: end for
```

2.4 Text document similarity

While in this chapter we have only focused on some specific techniques for assessing text document similarity, there are many other interesting approaches. In this last section we review some of the most popular methods.

The ability of assessing the similarity between texts is fundamental for a broad number of tasks, such as information retrieval [59, 91], document clustering [63, 133], text classification [7, 144], machine translation [110], or text summarization [42], among others. In order to compare texts, the similarity between words is what usually gets measured actually. This allows for comparing larger text units like sentences or paragraphs, since these could be seen as a combination of words, while semantic information which becomes lost at a lower level (character level) is still retained.

Words then offer the possibility of comparing texts at two different levels: lexical and semantic. On the one hand, two words are lexically related if they share the same sequence of characters (although they can refer to different things, e.g. *bank*). On the other hand, two words are semantically related if they have a similar meaning or refer to related ideas (although they can be represented by totally different sequences of characters, e.g. *lift* and *elevator*).

To compare words at the lexical level, the most straightforward approaches are based on comparisons of characters. The Longest Common Substring method (LCS) [58] determines the similarity between two words according to the number of characters contained in the longest common subsequence. Another possibility is not to look for subsequences but for characters in similar positions (e.g. *hand* and *hunt* have two characters, *h* and *n*, in the same positions), as the Jaro distance metric [75] does; this approach is extended in the Jaro-Winkler distance [146], which favours words that share the first characters. A third alternative is the Levenshtein distance [90], which is based on the number of changes required to turn one word into the other one, extended in the Damerau-Levenshtein distance [35] which allows transpositions to accomplish that task. These methods can be helpful when dealing with typographical errors [131, 35], and are commonly used in bioinformatics [57, 53]. Character-based approaches are also useful for information retrieval in Chinese or Japanese [44, 79].

However, these methods focus exclusively on the word as a set of characters and ignore important information, like the relative importance of a word within the text where it occurs, or the relevance of a word in a given set of documents, shortcomings which make them less suitable for information retrieval in general. Methods based on terms overcome that problem. In the same way that the previous methods focus on a character's role within a word, these methods mainly consider a term's role within a text[4]. Also, they

[4]For the sake of simplicity, in this section we consider each single word to be a term;

usually invoke the bag-of-words model, where a document is represented as an unordered collection of words.

The most basic methods of this kind are based on the boolean model [123]. In this model, a document is represented as a set of terms, and queries are boolean expressions connected by *AND*, *OR* and *NOT*. The documents are then retrieved depending on whether or not they contain the query terms. This model has many limitations nevertheless. A document either matches the query or not, so there are no partial matches, and as a consequence often too many or too few relevant documents are retrieved. Also, since all documents are equally relevant or irrelevant, it is hard to rank the retrieved documents unless additional information is available to use it as ranking criterion (e.g. other users' ratings, number of comments, etc.). Finally, it is difficult to represent a text document by means of boolean queries. Therefore, this model is mainly suitable for simple query-document similarity, but it is not an ideal choice for document-document similarity.

In order to speed up computations, in the boolean model documents can be seen as vectors of boolean values, where each component in the vector corresponds to a term and has a value of 1 or 0, depending on whether or not that term is contained in the document. This is a point in common with the more popular vector space model [124], where documents are also represented as weighted vectors, and which we have examined in detail in Section 2.1. Finally, we have the state-of-the-art alternative to the vector space model, language models [117], which we have reviewed in Section 2.2.

The main weakness of the previous approaches is that since they operate at the lexical level, semantic information is ignored. So when two words are semantically related but lexically different, they are not recognized as similar even if they really are, e.g. *truck* and *lorry*. On the other hand, homonyms are considered as similar even when they refer to different concepts (e.g. a *bow* to shoot arrows and the *bow* of a ship). To deal with this problem texts must be compared at the semantic level. To this end, several methods have been proposed, based both on the vector space model and probabilistic models.

In the vector space model, the most popular approach is Latent Semantic Analysis (LSA) [88], which is based on the idea that semantically similar terms occur in similar documents. By making a matrix that describes the occurrences of terms in documents (where rows correspond to terms, columns correspond to documents, and values are typically calculated applying the tf-idf weighting scheme) and applying singular value decomposition (SVD) to it, terms can be represented as vectors and then be compared in the vector space model. This approach has several points in common with the Hyperspace Analogue to Language model [99], although in this case two terms are semantically similar when they usually occur with the same words. The

for other possibilities we refer to Section 2.1.1

considered context when comparing two terms is then derived only from the surrounding terms and therefore, unlike the previous models, is not based on the bag-of-words model as word order is important. Other alternatives use external sources of information. For example, Explicit Semantic Analysis (ESA) [46], described in Section 2.1.3, calculates how related a document is to a given concept based on a given document collection, Wikipedia[5] in the original approach, or alternatives like the Reuters corpus of articles[6] [4]. A different approach is the normalized Google distance (NGD) [34], based on the number of documents found by Google for the potentially related terms where they occur both alone and simultaneously. On the other hand, several approaches have been proposed that use the relationships defined in the semantic networks available at WordNet[7] and MeSH[8] [140, 148, 59].

Finally, semantic information can also be taken into account using probabilistic models. From LSA evolved Probabilistic Latent Semantic Analysis (PLSA) [60], with a probabilistic grounding instead of based on linear algebra like LSA, and from this Latent Dirichlet Allocation [21], a generative model that allows to discover the latent topics underlying a document that is receiving much attention recently. We refer to Section 2.3 for more details on this approach.

[5]http://en.wikipedia.org

[6]http://trec.nist.gov/data/reuters/reuters.html

[7]http://wordnet.princeton.edu/, a lexical database of English where words are grouped into sets of cognitive synonyms, each expressing a distinct concept, interlinked by means of conceptual-semantic and lexical relations.

[8]http://www.ncbi.nlm.nih.gov/mesh, Medical Subject Headings, a hierarchically arranged thesaurus for biomedical literature.

Chapter 3

Information filtering

The goal of this chapter is to give an overview of the work that has been carried out in the field of information filtering regarding scientific resources, with a particular emphasis on research papers. We start the chapter introducing information filtering in general terms and reviewing the main approaches to this task. Then we focus on how information filtering has been applied to the recommendation of scientific resources. As we have said, we pay special attention to research paper recommendation, but we also review other interesting applications such as citation recommendation or expert finding. We also examine the repositories on which information filtering is applied, such as Current Research Information Systems (CRISs) and digital libraries. Finally, we analyze how some of the discussed methods are used in six popular systems.

3.1 Introduction

Due to the rapid increase in popularity of the World Wide Web in the last decades, the amount of information contained in it has long exceeded the limits of what users can handle. The need for some help to avoid drowning in such an ocean of data has contributed to the growing attention to information retrieval (IR) and information filtering (IF). Information retrieval as we know it nowadays originated in the late 1940s, when computerized methods started to be developed to deal with the considerable amounts of scientific information originated in those years [125]. Often used as synonyms, IF has many points in common with IR, but also differs from it in several aspects [16, 56]. First, IF systems are designed for regular users with long term needs and repetitive usage, while IR systems focus on satisfying a one-time information need at a given moment. This is the reason why user needs in IF are modeled by the system and kept in the form of user profiles, while IR systems do not usually know anything about the user and a query suffices to describe his information need. Also, IR systems select from a database those relevant data that match a query, while IF filters out irrelevant data from an incoming data stream, or collects relevant data from certain sources, always according to the user's profile.

In the aforementioned characterization of IF systems we can recognize features of what we typically know as recommender systems. The reason is that recommender systems are a specific type of IF systems, namely active IF systems [56]. In this case, the system searches a specific space collecting relevant information to the user, according to the interests described in his profile. This relevant information is then presented to the user. Therefore, the system needs to "act": first by searching and then by offering the information. The opposite to these active recommenders are passive IF systems. In this case, the system could be seen as a kind of barrier between the user and the data stream, letting only the data that match the user's profile pass. Although properly speaking these systems are information filters rather than recommender systems, the differences are few and they are often labeled as recommenders as well. In the remainder of the chapter we therefore make no differences between recommenders (active filtering systems) and information filters (passive filtering systems).

Recommenders have obtained a lot of attention in the last years. On the one hand, recommenders are appreciated by users since they help them to satisfy their information needs without having to dedicate too much time to search or to browse a whole site. On the other hand, from a commercial point of view, recommenders are also attractive as they are not only an added value for the users, which may lead to a higher number of customers, but these customers are also presented with potentially interesting items, which in many cases leads to more sales. As a result, nowadays it is possible to find recommenders on the Web applied to many different domains: shopping (Amazon[1]), films (NetFlix[2]), music (Last.fm[3]), books (GoodReads[4]), news (News360[5]), or scientific resources, on which we will focus in the next sections of this chapter. Depending on the purpose, and therefore on the information used, the recommendation methods vary, but most of them can be classified into three main categories: content-based filtering, collaborative filtering, and hybrid approaches. In the remainder of this section we introduce some basic concepts about these approaches, to end with a fourth category in which we briefly review other, less popular methods.

[1]http://www.amazon.com
[2]http://www.netflix.com
[3]http://www.last.fm
[4]http://www.goodreads.com
[5]http://www.news360.com

3.1.1 Content-based filtering

In content-based recommender systems, the items to be recommended are represented by a set of features based on their content [97]. For example, in news recommendation the features which describe an item can be derived from the title and body of a news article, while in movie recommendation these features can be actors, plot, genre, etc. In most content-based recommenders, even if the item itself does not consist of text, the features describing it are usually derived from textual content, as in the movie example. On the other hand, a user is represented by his profile, which can vary from a list of keywords to a list of items that represent his interests best (e.g. a list of movies that he has watched before). Since a user's interests can change in time, his profile can be updated, explicitly by the user or implicitly learned from his behavior over time [1].

The representations of the items are then compared to the user profiles using different approaches. Most content-based recommenders use relatively simple retrieval models, such as keyword matching or the vector space model with basic TF-IDF weighting [97]. Examples of such systems are [93] or [105]. However, as we saw in the previous chapter, these methods ignore semantic information, and therefore other approaches are sometimes used to tackle this problem, like in [41] or [36], which use information from WordNet to add extra linguistic knowledge. An alternative to these approaches, which are closer to IR, are machine learning techniques. In this case, the system learns the user profile, and according to that information it classifies items as interesting or not. The methods used in these recommenders are mostly based on naïve Bayes classification [114, 17] or relevance feedback and Rocchio's algorithm [127, 9].

One of the advantages of content-based recommenders is the fact that, unlike in those based on collaborative filtering, recommendations for a given user do not depend on other users' ratings, which is important as explicit ratings by other users are not always easy to obtain. This also allows the system to recommend new items that nobody has rated yet, which is a typical problem in collaborative filtering. Finally, these recommenders are also more transparent, as in many cases it is easy to list the features that influenced a recommendation and the user can use this information to decide whether to trust it.

However, these systems also have some drawbacks. Since they always match the items against the same user profile, the recommendations will always be similar unless the profile is updated or new items become available. This disadvantage, also referred to as the serendipity problem, makes it hard for the user to explore new types of items. Also, some representations cannot capture all aspects of the content, thus ignoring some aspects that could actually influence the user. For instance, a movie can be represented by actors, director, and genre, but there are many more factors that influence liking a movie (e.g. the pace, the music, the photography, etc.).

For further information on content-based filtering we refer to [97].

3.1.2 Collaborative filtering

Collaborative filtering (CF) is the process of filtering or evaluating items using the opinions of other users [126]. In these systems, the user gets recommendations of items that are liked by users whose preferences are assumed to be correlated with the user's preferences. To indicate the interest of each user in a given item, a rating is used; for example movies can be rated by giving a score ranging from zero ("did not like at all") to five ("loved it"). Alternatives to these scalar ratings are binary ratings, for example to indicate like/dislike, or unary ratings, to indicate that the user has observed/purchased/liked an item, and where absence of rating indicates no information about the relation between that user and that item. These ratings are usually stored in a matrix with as many rows as users and as many columns as items, and a system may use more than one matrix simultaneously. An online store, for example, could work with three matrices: one with scalar values, storing the ratings given by users to the items, another matrix with unary ratings to indicate which items the user has purchased, and a third matrix with unary ratings to indicate that the user has browsed some items.

According to [27], CF algorithms can be divided into two classes: memory-based algorithms, and model-based algorithms. Memory-based algorithms simply store all the ratings and use that information directly to make predictions. This kind of algorithms can also be subdivided into two categories: user-based algorithms and item-based algorithms. In user-based algorithms, the ratings of each user u are compared to those of the rest of the users, usually by means of Pearson correlation or cosine similarity [27]. If a user u' has given similar ratings to the same items as u, the system concludes that u and u' are similar, and it will consider those items rated highly by u' as potentially relevant for u. On the other hand, item-based algorithms make their recommendations based on the similarity between items: all ratings given to an item i are compared to those of the rest of the items, again using the Pearson correlation or cosine similarity. If i is usually rated similarly as an item i', they are considered as similar. Now, if a user has rated i but not i', his rating for i' can be predicted by looking at his rating for i (and for the other items similar to i' for which his ratings are available). If this rating is high, the item might be recommended to him.

Unlike memory-based algorithms, model-based algorithms do not consult the user database each time a recommendation must be made, but they use it to estimate a model that is then used to make predictions. These models can be estimated in different ways, such as using cluster models or Bayesian networks, as proposed by [27], or latent factor models [151, 2]. The idea behind these models is that there are a number of factors which are very specific for a given domain and which are difficult to measure (e.g. the complexity of the characters in a film or a book). However, these factors may influence the ratings and should therefore be taken into account as well. Similar to the LDA approach discussed in Section 2.3, these latent features can be discovered and used by specific probabilistic models. This kind of models can be found in [61], which uses PLSA, or [84].

CF algorithms have several advantages. On the one hand, they allow users to discover items different from those in their history. For example, a user of an online shop who has only bought books so far, could get interesting recommendations about films just because other users who liked those books also liked those films. With content-based filtering this is not always possible. On the other hand, the user ratings allow to recommend items of the specific quality that match the tastes of the user. For instance, two users u_1 and u_2 might be interested in horror films but this does not mean that they should get the same recommendations: u_1 might be interested in blockbusters while u_2 is interested in B movies. Users with the same affinities as u_1 will probably rate other blockbusters higher than any B movie, and users similar to u_2 will do the contrary, so both u_1 and u_2 will get the right recommendations. Also, CF algorithms can filter any type of content, regardless of how complex it could be to represent it, since representations of the content are not needed.

CF algorithms have some disadvantages nevertheless. The most common one is probably the cold start problem. Since recommendations are based on past ratings, when a user is new to the system there is no knowledge about him, and therefore no accurate recommendations can be made until the user has rated a sufficient number of items. Something similar happens to new items: they must be rated by a sufficient number of users before they can be correctly recommended to other users. Also related to this fact, sometimes it may happen that popular items, which have been more frequently rated, are more often recommended, leaving less room for similar but less known items, and so impacting the diversity of recommendations negatively. Finally, another typical problem are the so-called gray sheep, users with tastes that do not really fit in any group and whose ratings are therefore hard to predict. Also, a step further we find the black sheep: users with really particular tastes for which recommendation is nearly impossible.

More detailed information about collaborative filtering can be found in [126, 84].

3.1.3 Hybrid approaches

As we can see, both content-based and collaborative filtering algorithms have important advantages, but also drawbacks. For this reason, hybrid recommenders have been proposed to combine both approaches, which in theory would sum the strengths of both methods while minimizing the weaknesses. For example, the cold start problem of collaborative filtering would be mitigated since in those cases the recommendation could be made based on the content, and the serendipity problem associated to content-based algorithms would also be attenuated.

Most hybrid approaches focus on how to combine different existing methods for content-based and collaborative filtering. The simplest way is just displaying the results from both methods in a single list of recommendations (ranking them based on the predicted ratings, for example) [130] or combining the scores from the separate recommenders by calculating a weighted average score for each item [104]. Another simple alternative is to show the results of only one recommender depending on the situation: one set of results is chosen, but if the confidence on the results is not high enough, the next set is considered [139]. More elaborated approaches include feature combination, in which the features normally used by one approach are used as input by the other one [12], or feature augmentation, in which new features are created by the first approach to use them as input for the second one [109]. An alternative to this last method consists not in creating features that can be used as input for the second recommender, but in creating a model that the second recommender can use to make its own recommendations [8]. Finally, it is also possible to run the recommenders sequentially, where the output of the first recommender is the input for the second one (i.e., the second recommender refines the results of the first one) [30]. For an in-depth analysis of hybrid approaches we refer to [31].

It is worth to note that although hybrid systems should produce better results than those based on one kind of approach only, this does not seem to be always the case, which is usually attributed to a bad performance of content-based filtering due to low quality metadata [22].

3.1.4 Other approaches

In this final section we consider other approaches which fall outside the previous classification. The most important recommenders within this category are knowledge-based recommenders [29]. These recommenders typically use some kind of constraint-based reasoning, where recommendations are made when certain constraints are satisfied, or case-based reasoning, where recommendations for previous, similar cases can be used to generate recommendations for a specific new case. Specific knowledge on the considered domain is usually needed in these cases, to define the set of constraints to be satisfied or the rules the system should follow. Another possibility is trust-based recommendation [5, 100]. In this case, the system makes use of a trust network to encode how much a user trusts the others. The idea is not to search for similar users as CF does, but to search for trustworthy users by exploiting trust propagation over the network. Then, the items appreciated by these users are recommended to the active user [100]. Finally, demographic recommenders use demographic information about the users, like age, gender, education, and other personal data to categorize them and then make recommendations according to each group [112].

With the appropriate knowledge base, these systems can provide valuable recommendations. However, this strength also relates to a major drawback: a costly knowledge engineering process is required. This is the main reason why other approaches like those from the previous sections are usually preferred. Also, the results can vary depending on the domain, something especially evident in the case of demographic recommenders. For example, it is not the same recommending movies which often have a very specific demographic target, than recommending research papers which usually are equally interesting for a whole scientific community.

3.2 Scientific information filtering

Scientific information is not an exception to the aforementioned explosive information growth. This has motivated the development of new tools to facilitate research-related tasks and the application of information filtering techniques. In this section we first briefly review some systems which gather and handle this information, and then we focus on methods designed to filter specific kinds of data. In particular, we dedicate most of our attention to research paper recommenders.

3.2.1 Managing scientific information

First we consider Current Research Information Systems (CRISs). These systems store information about current research being carried out by organizations and researchers. This information ranges from information linked to the nature of research being carried out, such as data, software, bibliographies, or results, to information related to the management of research, such as project descriptions, funding programmes, patents, or market/trend reports. We find examples of these systems operating at national or regional level in the USA[6], Norway[7], the Czech Republic[8], Flanders in Belgium[9], or Andalusia in Spain[10]. A detailed view on the history and current situation of CRISs can be found in [119]. The amount of information contained in these systems is considerable (e.g. information concerning several research institutions and projects of a whole country), and therefore they usually incorporate some search engine. Also, some of them follow specific standards

[6]http://cris.nifa.usda.gov
[7]http://www.cristin.no/english/
[8]http://www.isvav.cz
[9]http://www.researchportal.be
[10]http://sica2.cica.es

(e.g. CERIF[11]) which incorporate, among other things, semantic metadata that allow for better search possibilities [96].

However, it is interesting to observe that while information filtering or recommendation would certainly be helpful in this context, to our knowledge no attempts have been made to use a recommender in these systems, with the only exceptions of [66] and [67] where we addressed the problem by applying ideas from fuzzy-rough set theory. In particular, [66] describes an alert system to match users with potentially relevant project descriptions or funding programs. In this system both users and documents have a profile based on keywords, some of them selected from a specific taxonomy, while others can be manually introduced by the user or indicated by the author of the document, and then are added as a new level of the taxonomy. Additionally, the profiles indicate the relevance of every keyword for that user/document with a value between 0 and 1; [67] focuses on how to assign keywords and weights automatically. On the other hand, weights between 0 and 1 are also assigned to every relation in the taxonomy, representing how related two keywords are. The idea is then to enrich the profiles based on the relations in the ontology to subsequently assess their similarity.

On the other hand, more specialized than CRISs and also much more popular are digital libraries dedicated to scientific literature. These libraries can be interdisciplinary, such as ScienceDirect[12], or they can focus on a specific domain, such as PubMed[13], for medical literature, the ACM Digital Library[14] for computing, or IEEE Xplore[15] for technology in general. This kind of libraries usually include an advanced search engine that allows searching by multiple fields and, unlike CRISs, some of them do include recommendation functionalities, as we will see in detail in Section 3.3. Also, these libraries are usually the source of the information filtered by many of the proposed algorithms for paper recommendation which we review in the next section, and the search space of specialized search engines such as Google Scholar[16] or Microsoft Academic Search[17], whose recommendation features will also be reviewed in Section 3.3.

3.2.2 Research paper recommendation

Although the aforementioned systems are an important first step in separating relevant, research-related information from the rest of data available

[11]The Common European Research Information Format (CERIF) is a standard for CRISs proposed by the European Comission and currently managed by euroCRIS (http://www.eurocris.org)

[12]http://www.sciencedirect.com

[13]http://www.ncbi.nlm.nih.gov/pubmed

[14]http://dl.acm.org

[15]http://ieeexplore.ieee.org

[16]http://scholar.google.com

[17]http://academic.research.microsoft.com

in the Web, users still have to spend valuable time browsing through the systems searching for the desired information. To facilitate this process, several research paper recommendation techniques have been developed.

Most research in this area has focused on personalized recommendations: the system profiles the user based on his interests, and then presents him with a set of articles which might be relevant to him, usually ranked from more to less relevant. Users' preferences can be obtained both in an explicit or an implicit way. In the first case, the user can express his information needs by entering a set of keywords or selecting them from a given list [25], or indicating one or more papers that best represent his research interests. To obtain the user's preferences in an implicit way, the system can consider the papers written by the user as representative of his interests [11] or, when the user maintains a private library of papers relevant for his research (e.g. bookmarks of relevant papers in the field which he might usually consult), these papers can also be used [23]. In addition, the behavior of the user can be monitored, including which searches he has performed, which articles he has read, etc. [15]. Monitoring is also useful to implicitly obtain feedback about the articles recommended by the system, for example depending on whether the user clicked on the recommendation, or how much time he spent reading a recommended article.

Although the main goal of personalized recommendation of research papers is to avoid that researchers have to spend time looking for relevant articles, an interesting application is the so-called reviewer assignment problem (also called conference paper assignment problem), which consists in matching papers with reviewers according to their expertise when setting up the paper reviewing process for a conference. This specific problem has been addressed several times since the first approach by Dumais and Nielsen [40], who use Latent Semantic Indexing to match papers and reviewers whose expertise is given by their own abstracts. The abstracts written by the reviewers are usually the source of information about the reviewers' expertise, as in [150] or in [11].

However, many efforts are not dedicated to personalized recommendations, but rather to assist the user during the search process. In particular, they focus on searching similar or related papers to the one being browsed by the user, to offer them as potentially interesting papers (e.g. [78, 52]). This is the type of information filtering used in most non-experimental scientific libraries which offer some kind of recommendation, as we will see in Section 3.3, probably because it does not require the user to even register at the site. Also, this form of filtering can be used as basis for personalized recommendations, since as stated before the user profile can be seen as a set of papers (written or selected by the user), and therefore any paper related to the papers in that set would potentially be relevant to the user. Finally, it is worth to note that although this "related articles" feature is the most popular one when enhancing search engines, there are other approaches like the one in [3], which uses collaborative filtering to improve the results based on searches made by similar users.

3.2.2.1 Content-based methods

We first discuss the application of content-based methods to the problem of recommending research papers. These methods are solely based on textual information about the papers and the users. In the case of the papers, the only information used is their content, while in the case of the users there are more possibilities, with the usual ones being some keywords or categories selected/entered by them, the content of their own papers or other papers that they have selected, their emails, or their personal websites. It is important to note that with "content of research papers" we do not only refer to the body of the article, but also to all kinds of information usually available along with it, such as authors, keywords, journal, bibliography or cited and citing papers. Also, while some approaches use only one kind of information (e.g. only the abstract), other approaches combine different features.

A good example of this is CiteSeer[18] [25, 26]. On the one hand, CiteSeer allows the user to specify constraints to define the papers he is interested in, such as specific keywords that should occur in the text of the paper; on the other hand, the user can also select a set of papers in which case the system searches for other papers similar to those in the set. In the case of constraints, to assess the similarity between a user and a paper, the system simply checks whether the keywords occur in the document or not, while in the second case the popular combination of tf-idf and cosine similarity are combined. Citations are also an important feature in this system, since a notion similar to bibliographic coupling is used and therefore if two publications from the set selected by the user cite the same articles they are considered to be related. It is interesting to note that all types of information can be used simultaneously: the different models are compared and then the results are combined assigning more or less weight to each feature. These weights are adjusted not only manually by the users, but also automatically by monitoring their actions while they are logged into the system.

The method proposed in [136] also uses citations but in a different way. In particular, given a paper p, not only a term vector is made for p, but also for the articles that cite p and the articles cited in p. The profile of each paper is then made by combining its own term vector with those of the linked papers, while the profile of each user is created by doing the same for all the papers that he has written. The work explores several methods to include such contextual information in the profile by weighing it depending

[18]Do not confuse with CiteSeer^{X}, which replaced the original CiteSeer but is quite different.

on different factors: cosine similarity between papers, publication year, etc. The authors also discuss the view that junior researchers should probably be profiled in a different way than senior ones, due to the low number of papers they have written.

A similar approach is followed in the content-based methods studied in [138]. To search papers related to a given paper p, not only the abstract of p is used, but also the abstracts of the papers cited in p. This can either be achieved by computing the vectors independently and finally combining the similarity scores, or by concatenating the abstracts of p and those of the cited papers, and treating it as a single document.

A set of related papers is also used in [108], again with the aim of searching related papers given another paper. However, instead of using the citations to link papers as in the previous approach, the proposed method first extracts a list of technical terms from the paper, to then use them as a query to perform a search for papers. A set of papers that match the query is then retrieved and ranked with the HITS algorithm [83].

This way of working in two steps, where a set of candidate papers is first retrieved by using a specific method and then a second method is used to rank them, is not uncommon. For example [78], which also focuses on recommending papers similar to a given paper p, first uses the paper citation graph to narrow down the scope of candidates, which also helps to speed up computations. In particular, this citation graph consists of the papers cited in p, plus the papers that cite them, plus those that cite p, plus the papers cited in this last group. The candidates are then ranked based on their similarity with p. To this end, topic modeling is used, and the document-topic distribution for each document is estimated with Latent Dirichlet Allocation. The resulting models are compared in the vector space model: each document is then represented as a vector where each component corresponds to the probability of a latent topic. An interesting idea in this work is the fact that it does not work with the whole abstract. Instead, the abstract is first split in two parts, to exploit the typical structure of abstracts, where the problem is first explained and the solution is then introduced. Each paper is then represented by two documents: the part of the abstract related to the problem, and the part of the abstract related to the solution. This distinction results in two different recommendation lists, a problem-oriented one and a solution-oriented one, which offer the user a more specific choice depending on what he found interesting about paper p in first place (e.g. if he was browsing the article *Finding similar research papers using language models*, was he actually interested in research paper similarity, i.e., the problem, or in language models, i.e., the solution?).

3.2.2.2 Collaborative filtering methods

As mentioned in Section 3.1, the common alternative to content-based methods are those based on collaborative filtering. Since CF is based on users rating items, these methods are used to recommend papers to users more often than to suggest papers related to a given paper. This is for example the case in [111], which uses papers bookmarked by the users in CiteULike[19]. Since CiteULike allows users to provide ratings and tags, these are also taken into account to compute the similarity between users and to then rank the obtained papers (which had first been explored in [23]). In particular, three different methods are proposed: one based on the Pearson correlation over the ratings of the users' common papers (which is often considered as the "classic" CF method), a variation on the former that takes into account the number of raters for the ranking (i.e., papers rated by several users are ranked higher), and one based on the BM25 algorithm that uses the tags given by the users to the papers. An interesting observation made in this work is the fact that many users simply ignore rating or tagging the papers. Moreover, those users who do rate or tag sometimes have different rating criteria. Therefore, the information about tags and ratings, while useful, should be used carefully.

A different possibility is to mine the citations between papers and use them as ratings, in such a way that a citation to a given paper is interpreted as a positive rating for it; the ratings are then implicitly given by the users when they cite a paper in their own work. This idea is used for example in [138] and [102], which also propose alternatives to the classic CF method. In particular, [138] also uses the citations in the cited papers (i.e., a user has not only "rated" the papers cited in his work, but also those cited in those papers), while [102] explores approaches which are completely different to the classic method. On the one hand, a naïve Bayes classifier where co-citations are the positive training examples is used; on the other hand, the authors propose Probabilistic Latent Semantic Indexing, where the user gets recommendations about the papers with the highest probabilities relative to the latent classes with which he is related the most. However, these novel approaches did not seem to work very well. Alternatively, the previous idea of using citations as ratings can be polished and, instead of using boolean ratings, more complex ratings can be given, like PageRank scores computed out of the citation graph of the papers in the collection [141], although the quality of the recommendations actually seems to decrease when this ranking method is used.

Finally, another way to implicitly obtain ratings is monitoring the user. In [3], which focuses mainly on the performance of the CF algorithms, this idea is followed. Specifically, the system observes which papers the users access, and assume their interest in those papers. Moreover, users who ac-

[19]http://www.citeulike.org

cess the same set of papers are clustered, in what could be seen as research groups. The idea is then both to improve the performance (the number of "users" in the CF matrix is reduced) and to present the user with recommendations based on the interests of his research group.

3.2.2.3 Hybrid methods

In Section 3.1.3 we have seen that there are approaches that combine content-based with collaborative filtering techniques. The application of these hybrid approaches to research paper recommendation has also been studied in a number of works. An example of such a hybrid system can be found in [106], although it focuses more on the collaborative filtering part. In particular, the idea is to first filter the papers following a content-based approach, where a series of search words are matched to the content of some fields describing the paper like keywords, title, or language. The resulting set of papers is then ranked according to the scores obtained with collaborative filtering ("classic" CF). Interestingly, in this approach users do not rate each article as a whole, but they rate different points about the article, like originality, readability, literature review, etc., and the matching of users in the CF process is based on these separate ratings.

Torres et al. [138] study several hybrid approaches by combining the results of two independent recommenders: a content-based one and a CF one. Specifically, they use basic approaches for these two modules and focus on how to combine them, which can be done by running them sequentially (i.e., the output of the first one is the input of the second one, as in the previous example), or by running them in parallel and combining the resulting rankings. In particular, papers occurring in both rankings appear at the top of the final ranking, where their score is the sum of their ranks in the two separate rankings. Papers that only appear in one of the two separate rankings are appended next in the final ranking. It is interesting to see that hybrid approaches where the recommenders run sequentially do not perform very well in general, while combining the final results of both does seem to bring some improvement.

A similar approach is followed in [6]; in order to combine the content-based method (tf-idf combined with cosine similarity) with (classic) collaborative filtering, the final score for the similarity is obtained by computing a weighted average of the two separate results. More than in the recommendation methods themselves, the strength of the proposed digital library lies in its structure in folders. Like in a file system, it is possible to have a folder for a specific community, which contains a folder for each user, which in turn contains the papers he is interested in, maybe also subdivided in folders classified by topic. Such a structure allows to identify a user with a folder and compare it to the papers contained in other folders, but it makes it also possible, for example, to compare folders corresponding to users to find researchers with similar interests and so boost collaboration, or to compare a user's folder to that of a community.

Scienstein, the prototype proposed in [50], also uses a weighted average to combine the scores of the different methods, although in this case more than two methods are combined. Specifically, Scienstein uses several types of information, namely text, author, source (e.g. journal), ratings, and references, and different methods are used for some of these types: content-based methods for the three first types, collaborative filtering for the ratings, and techniques from citation analysis for the references. Each of the five methods gives a similarity score and, as mentioned before, a weighted average is calculated, with the special feature that the user can specify the weights to assign more importance to some specific types of information. Scienstein is now discontinued, although it has successors in SciPlore[20], which uses the same citation analysis techniques as Scienstein, and Docear[21] [15], which is still in development.

Lastly, it is also posible to combine two approaches without doing it sequentially or combining the results at the end, as in the previous methods. For example, [143] proposes to combine collaborative filtering based on latent factor models with topic models, resulting in the collaborative topic regression model, which represents users with topic interests and assumes that documents are generated by topic models. On the one hand, latent factor models use information from a user's library, which makes it possible to recommend articles from other users who have liked similar articles. On the other hand, topic models are based on the content of the articles, which makes it possible to recommend articles that have not been rated by any user yet, something that would not be possible with the latent factor models alone. This is a good example of what hybrid systems pursue, i.e., combining the different strengths of both content-based methods and collaborative filtering.

3.2.2.4 Citation analysis

During the study of the different existing approaches to research paper filtering and recommendation, we have observed that citations play an important role in a substantial number of methods, forming almost a separate category on their own. This is why we dedicate this last subsection to introduce some concepts related with these techniques and to comment some solutions which use them.

A central part of many techniques related to citations is the citation graph. This is usually a directed graph where the nodes represent the papers and the edges represent the citations. Most systems work with the whole graph, although some methods take only into account the closest

[20]http://www.sciplore.org
[21]http://www.docear.org

neighbours, e.g. [78]. The citation graph is mainly used for two purposes in the task of recommending research papers. On the one hand, as mentioned in Section 3.2.2.2, it can be considered as a way to implicitly rate papers, where each paper "rates" the papers it cites, and where it gets "rated" by the papers citing it. This is not only an easy way to obtain ratings, but it also ensures a high number of them, since all papers cite other papers (while in a system where users rate, many users often do not rate any items at all) [138, 101]. On the other hand, due to its analogy to hyperlinks on the Web, ranking algorithms originally designed for the Web can thus be used to rank scientific articles. In particular, many approaches explore the use of the popular PageRank algorithm [28] either in its original form [52, 141] or with some modifications [85], with HITS being an alternative [108].

However, although all citations are represented equally in the citation graph, there are differences among them that cannot be captured by such representation. For example, Liang et al. [92] define three different types of citations: when the paper is somehow based on the cited paper, when the paper and cited paper try to solve the same problem but use different methods, and when the cited paper is mainly intented to introduce some background information. Depending on the type of relation between papers, the edges of the graph can get a higher or lower weight, which seems to improve the results. Also, Gipp and Beel [49] study new ways to exploit citations to recommend papers. Apart from the well-known bibliographic coupling ("two documents citing the same documents are likely to be related") [82] and co-citation analysis ("two documents usually cited together are likely to be related") [129], they propose citation proximity analysis and citation order analysis. The first one is a refinement of co-citation which does not only look at the citations, but also at their position in the text. The idea is that two papers which are cited, for example, in the same sentence, are probably more related than two papers cited in different sections of the paper. This approach has proven useful and actually is the technique followed by the previously mentioned SciPlore. On the other hand, citation order analysis is related to bibliographic coupling, but also taking into account the order on which papers are cited in different documents: two documents which cite the same documents in the same order are probably more related than if the common citations appear in different order. Moreover, this has other applications than paper recommendation: if the same sequence of citations is found in different papers, it can point to potentially plagiarized work.

3.2.3 Academic recommendation

Most recommendation methods for academic resources focus on recommending research papers, since research papers are arguably the main way of knowledge transfer within academia and the main type of content in the dedicated digital libraries. However, it is worth to mention that paper recommendation is not the only application of recommenders in academia.

Closely related to paper recommendation is citation recommendation. Actually, it can be seen as a particular case of paper recommendation in which papers are identified which are similar to a given paper, so the strategies are similar to those used for paper recommendation. For example, [101] uses the previously seen collaborative filtering approach where the citations in every paper are used as ratings for the cited papers; each paper is therefore seen as a "user" that rates other papers, and the recommendations are made based on that information. A content-based approach is used in [134], which proposes combining different content features with the citation graph. In particular, for a given paper the system measures the similarity between texts to retrieve a first set of papers, which is further expanded with the papers cited in those. All these papers are then ranked based on information that can be retrieved from the citation graph such as co-citations, citation count, or Katz distance between nodes. Nallapati et al. [107] also use both text and citations, but in a different way. More specifically, their idea is to build a joint model based on text and citation in the topic modeling framework, combining LDA and PLSA into a single model called Link-PLSA-LDA. Finally, Huang et al. [64] do not focus on the citations themselves, but on the context of the citations, i.e., the text around the citation where the author explains the content of the cited work. Based on those terms, they create a translation model which, given a specific query, returns a list of possible works that could be cited.

Also, recommendation of researchers with similar interests can be useful and has been explored in some works. Xu et al. [147] propose a method based on both social network analysis, to deal with the relationships between researchers, and semantic analysis, based on a domain ontology, to analyze the semantic similarity of researchers' expertise. The approach in [32] is based on network analysis too, although in this case it is a co-autorship network, and the methods measure the similarity between two researchers depending on their distances in the graph and on common co-authors. On the other hand, [51] proposes a completely content-based approach which either models researchers from their papers to compare them, when probabilistic modeling methods are used, or treats one profile as a query and the other as a document, when using the vector space model. Finally, [19] presents a method not oriented to actual researcher recommendation but somehow related to it. In particular, they propose to use personal agents to help researchers in their search for relevant information, like scientific papers. However, these agents do not only help searching: they learn from the user's behavior. The idea then is that agents belonging to senior researchers can share their information with those of novices, which actually results in senior researchers sharing their expertise with novice ones.

3.3 Research paper filtering systems

We end this chapter reviewing some popular research paper search engines, focusing on their recommendation features, to see how some profesional, well-established systems start to incorporate methods like the ones commented in Section 3.2.2, which until recently were limited to experimental prototypes.

3.3.1 Google Scholar

Google Scholar[22], released in 2004, is a subset of the larger Google search index, consisting of full-text journal articles, technical reports, preprints, theses, books, and other documents deemed to be scholarly [142]. Similar to the Google main search engine, it has a simple interface consisting of a text box to enter the query. The system then retrieves a list of potentially relevant documents ranked, according to its developers, "in the way researchers do, weighing the full text of each document, where it was published, who it was written by, as well as how often and how recently it has been cited in other scholarly literature". While the details of this ranking algorithm are unknown, research by Beel and Gipp [13, 14] showed that articles' citation counts have a significative impact on the ranking. The article's title also seems to play an important role in the algorithm, so articles whose titles match several query terms tend to be ranked higher. However, the presence of the same terms in the full text has a lesser impact, and their frequency of use seems to be ignored.

The reason why we include Google Scholar in this list of recommenders is the recommendation service included in August 2012. To use this feature, the user must first create a profile, which basically includes his articles, automatically retrieved by the system, although they can also be selected manually. This profile consisting of the user's articles is then used to filter the new articles arriving to the system, and the potentially relevant ones are selected and presented to the user.

Like the ranking algorithm of the search engine, the recommendation algorithm used for this feature is unknown. However, as the filtered articles are all recent, no citation information is available yet, and therefore this information loses relevance here. A reasonable possibility, looking at some personal recommendations, is that the most relevant terms are first retrieved from the user's articles (mainly from title and abstract) to subsequently use them as input for querying the system's index. The approach seems therefore to be content-based, although probably other information such as reputation of the authors is taken into account, i.e., papers of authors with many citations are probably ranked higher. Also, it is interesting to remark that in the user's profile it is possible to specify co-authors (some of them

[22]http://scholar.google.com

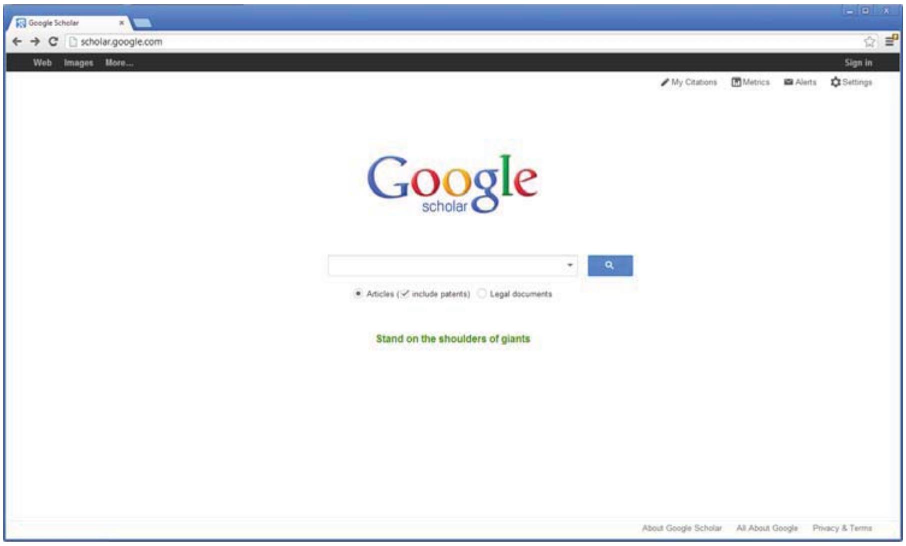


Figure 3.1: Google Scholar's homepage

are automatically added by the system), which could open the possibility for a collaborative filtering approach in the future. Also, this information can be used to build a kind of social network and recommend papers of known researchers (e.g. co-authors of co-authors).

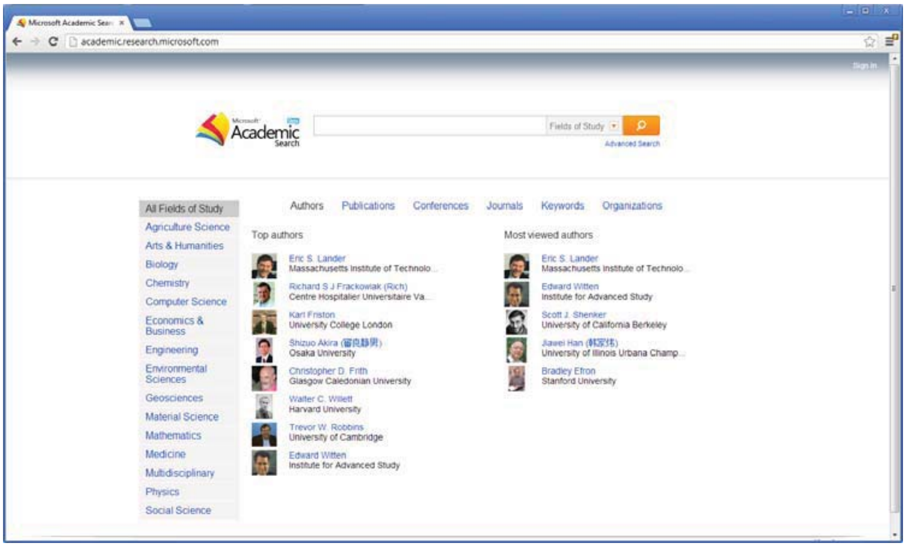
Finally, Google Scholar offers a "related articles" feature, which allows to obtain articles similar to a specific one. The strategy followed in this case seems to be similar to that for personalized recommendation, but with some differences. The main one is probably the fact that much more importance seems to be given to the author, which causes other articles by the same authors to be ranked high in the recommendation list. Moreover, the citations seem to gain importance, and while they seem to have less influence than on the search results, they still seem to be taken into consideration.

3.3.2 Microsoft Academic Search

Microsoft Academic Search[23] is Microsoft's search engine for academic publications. Relased in 2009, its interface is a bit more complex than Google Scholar's one. It features a text box to enter the query, but from the homepage it is also possible to directly browse data about authors, publications, conferences, journals, keywords, and organizations, although this is restricted to the top items of each category only; to find other items the search engine must be used. Similarly to Google Scholar, Microsoft Academic Search's ranking algorithm is unknown, although the developers

[23]http://academic.research.microsoft.com

claim that the results are sorted based on "their relevance to the query" (more likely matching the query to title and abstract) and on "a static rank value that is calculated for each item in the Microsoft Academic Search index. The static rank encompasses the authority of the result, which is determined by several details, such as how often and where a publication is cited".



Top authors

Eric S. Lander
Massachusetts Institute of Technolo...
Richard S J Frackowiak (Rich)
Centre Hospitalier Universitaire Va...
Karl Friston
University College London
Shizuo Akira (審良静男)
Osaka University
Christopher D. Frith
Glasgow Caledonian University
Walter C. Willett
Harvard University
Trevor W. Robbins
University of Cambridge
Edward Witten
Institute for Advanced Study

Most viewed authors

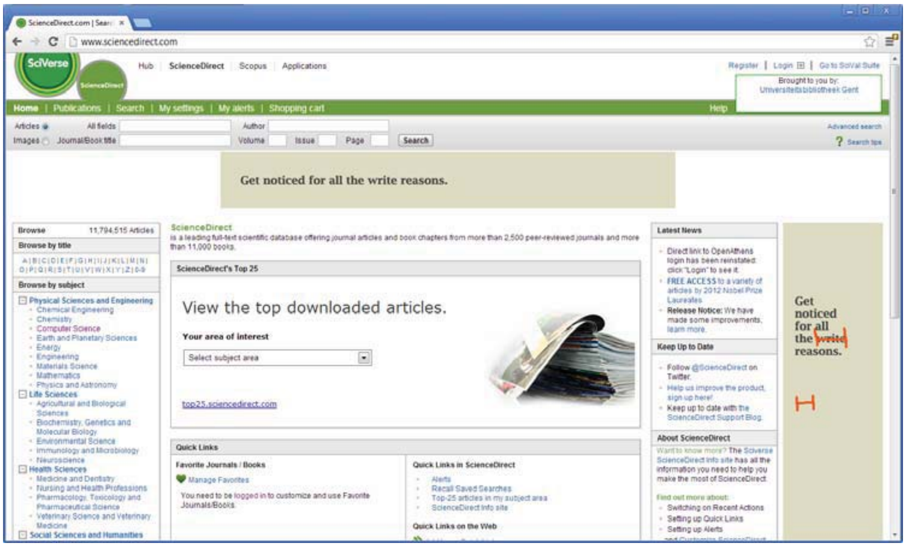
Eric S. Lander
Massachusetts Institute of Technolo...
Edward Witten
Institute for Advanced Study
Scott J. Shenker
University of California Berkeley
Jiawei Han (韩家炜)
University of Illinois Urbana Champ...
Bradley Efron
Stanford University

Figure 3.2: Homepage of Microsoft Academic Search

In general, Microsoft Academic Search offers more features than Google Scholar, making it possible to search and group by conference, journal, etc. It also offers some interesting visualization tools to work with the citation graph of a paper, or the co-author graph of a researcher, although it lacks a personalized recommender system like that of its Google counterpart. It does have a "related publications" recommendation feature, which is why we include this system here. Again, the recommendation algorithm used is unknown, and actually this feature is not available for all papers. However, it seems to be related to the number of citations, as generally this feature is not shown for papers with few citations and the number of "related" publications is usually larger for papers with many citations. This, along with the stress put on citations and graphs in other features of the system, seems to indicate that the recommendation algorithm is somehow related to the citation graph, although the actual method is not clear.

3.3.3 ScienceDirect

ScienceDirect[24] is Elsevier's digital library containing full text journal articles, and since 2010 it is integrated in Elsevier's platform SciVerse. The advantage of ScienceDirect over Google Scholar or Microsoft Academic Search is the fact that since it indexes Elsevier's own articles, users can access all full texts (if they have paid a subscription). However, as it only indexes own articles, its coverage is smaller than those other systems'.


ScienceDirect.com | Sear

www.sciencedirect.com

SciVerse ScienceDirect

Hub | ScienceDirect | Scopus | Applications

Register | Login | Go to SciVal Suite

Brought to you by: Universiteitsbibliotheek Gent

Home | Publications | Search | My settings | My alerts | Shopping cart | Help

Articles ● All fields [ ] Author [ ]
Images ○ Journal/Book title [ ] Volume [ ] Issue [ ] Page [ ] Search

Advanced search

? Search tips

Get noticed for all the write reasons.

Browse 11,794,515 Articles

Browse by title

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | 0-9

Browse by subject

- Physical Sciences and Engineering
  - Chemical Engineering
  - Chemistry
  - Computer Science
  - Earth and Planetary Sciences
  - Energy
  - Engineering
  - Materials Science
  - Mathematics
  - Physics and Astronomy
- Life Sciences
  - Agricultural and Biological Sciences
  - Biochemistry, Genetics and Molecular Biology
  - Environmental Science
  - Immunology and Microbiology
  - Neuroscience
- Health Sciences
  - Medicine and Dentistry
  - Nursing and Health Professions
  - Pharmacology, Toxicology and Pharmaceutical Science
  - Veterinary Science and Veterinary Medicine
- Social Sciences and Humanities

ScienceDirect
is a leading full-text scientific database offering journal articles and book chapters from more than 2,500 peer-reviewed journals and more than 11,000 books.

ScienceDirect's Top 25

View the top downloaded articles.

Your area of interest

Select subject area

top25.sciencedirect.com

Quick Links

Favorite Journals / Books

Manage Favorites

You need to be logged in to customize and use Favorite Journals/Books.

Quick Links in ScienceDirect

- Alerts
- Recall Saved Searches
- Top-25 articles in my subject area
- ScienceDirect Info site

Quick Links on the Web

Latest News

- Direct link to OpenAthens login has been reinstated: click "Login" to see it.
- FREE ACCESS to a variety of articles by 2012 Nobel Prize Laureates
- Release Notice: We have made some improvements, learn more.

Keep Up to Date

- Follow @ScienceDirect on Twitter.
- Help us improve the product, sign up here!
- Keep up to date with the ScienceDirect Support Blog.

About ScienceDirect

Want to know more? The SciVerse ScienceDirect Info site has all the information you need to help you make the most of ScienceDirect.

Find out more about:

- Switching on Recent Actions
- Setting up Quick Links
- Setting up Alerts

Get noticed for all the ~~write~~ reasons.

Figure 3.3: ScienceDirect's main page

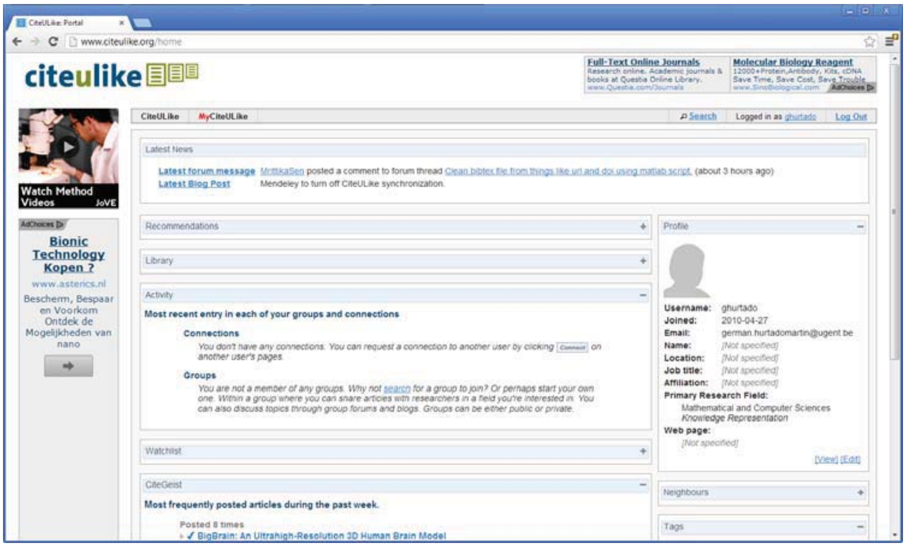
Since ScienceDirect is mainly a digital library it is slightly different to the previous systems, although it does have a search engine which allows users to search by field and also to formulate more complex queries by using boolean connectors. Any query made in this engine can also be saved and used for the system's alert service, which notifies the user when a new article matching the query is added to the database. Even more relevant is the "related articles" feature. The algorithm used is unknown, and no information is given about the data or methods used, other than "intelligent document matching". Looking at personal recommendations, the system seems to follow a content-based method where the citations play a secondary or no role. This would not be surprising, as the system has access to the full text of all articles and can therefore estimate quite accurate models. Also, working with documents that follow the same publisher-specific format makes it easier to extract and use extra information such as keywords given by the authors or categories in a particular classification.

[24]http://www.sciencedirect.com

3.3.4 CiteULike

CiteULike[25], created in November 2004, is not a digital library or a specialized search engine, but rather a social bookmarking site that allows the user to manage his library of research papers (own papers, papers he is interested in, papers he often references, etc.) while sharing that information with other researchers.

As a social bookmarking site, CiteULike's strength lies precisely in sharing. A user can share the articles in his library with a specific user or group of users, but he also can tag the articles, which can be helpful for other users to know what the articles are about, or can indicate his priority to read the article, which can give an idea of his interest in a given topic. This collaborative concept of CiteULike has made it an attractive system to explore collaborative filtering approaches [111, 22], and it also has its own recommendation mechanism based on this idea.


CiteULike: Portal
www.citeulike.org/home

citeulike

Full-Text Online Journals
Research online. Academic journals & books at Questia Online Library.
www.Questia.com/Journals

Molecular Biology Reagent
12000+Protein,Antibody, Kits, cDNA Save Time, Save Cost, Save Trouble
www.SinoBiological.com

AdChoices

Watch Method Videos JoVE

AdChoices

Bionic Technology Kopen ?
www.asterics.nl
Bescherm, Bespaar en Voorkom Ontdek de Mogelijkheden van nano

CiteULike | MyCiteULike | Search | Logged in as ghurtado | Log Out

Latest News

Latest forum message MrittikaSen posted a comment to forum thread Clean bibtex file from things like url and doi using matlab script. (about 3 hours ago)
Latest Blog Post Mendeley to turn off CiteULike synchronization.

Recommendations

Library

Activity

Most recent entry in each of your groups and connections

Connections
You don't have any connections. You can request a connection to another user by clicking Connect on another user's pages.

Groups
You are not a member of any groups. Why not search for a group to join? Or perhaps start your own one. Within a group where you can share articles with researchers in a field you're interested in. You can also discuss topics through group forums and blogs. Groups can be either public or private.

Watchlist

CiteGeist

Most frequently posted articles during the past week.

Posted 8 times
✓ BigBrain: An Ultrahigh-Resolution 3D Human Brain Model

Profile

Username: ghurtado
Joined: 2010-04-27
Email: german.hurtadomartin@ugent.be
Name: [Not specified]
Location: [Not specified]
Job title: [Not specified]
Affiliation: [Not specified]
Primary Research Field:
Mathematical and Computer Sciences
Knowledge Representation
Web page:
[Not specified]
[View] [Edit]

Neighbours

Tags

Figure 3.4: CiteULike's user page

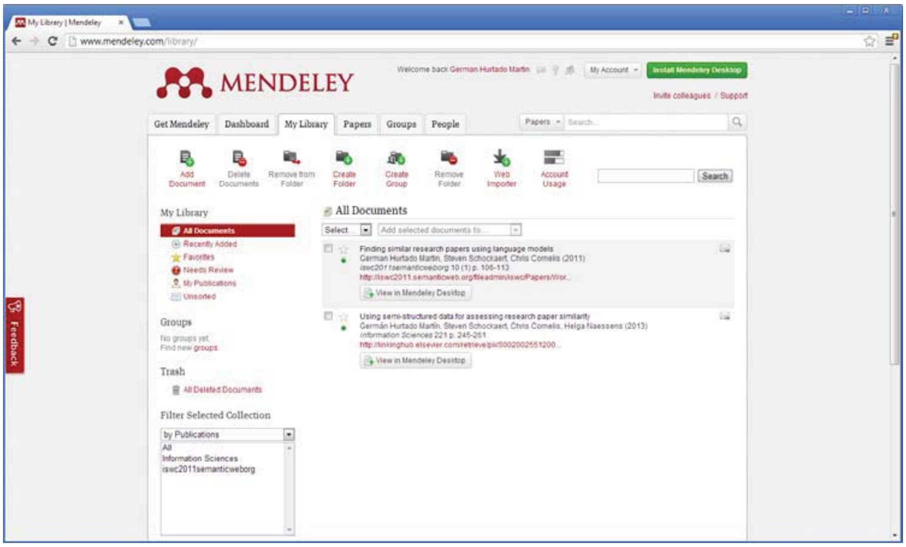
In particular, for users with at least 20 articles in their libraries (to ensure reasonable recommendations), CiteULike uses a CF algorithm, based on the papers that each user has in his library: if a paper is in a user's library it counts as a positive rating of that paper given by that user. The result is then a list of potentially relevant papers that can help the user to discover not only new papers but also like-minded users whose libraries might be interesting to follow. More refined CF techniques could include using the ratings that indicate the reading priority or comparing the tags [22].

[25]http://www.citeulike.org

Also, like the previous systems, CiteULike includes a "find similar" function that allows to find articles similar to the browsed one. This function is based on content-based filtering using title and abstract. Specifically, it seems to extract some terms from the given paper and use them as query to search for similar papers. Although this can lead to include too general terms to the query, an interesting feature is the fact that the terms used in the query are actually shown to the user, so it is clear why he has received some specific recommendations. It is also worth to mention that this function does not limit itself to search for similar papers, but it also uses the query to search for similar users and groups.

3.3.5 Mendeley

Mendeley[26], released in August 2008, is an application for managing and sharing research papers. The system actually consists of two parts: a desktop application, with which the user can manage the research papers on his hard drive, and a web application which allows, among other things, to share the documents with other users and to access a social network.



Finding similar research papers using language models
German Hurtado Martin, Steven Schockaert, Chris Cornelis (2011)
iswc2011semanticweborg 10 (1) p. 106-113
http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/Wor...
View in Mendeley Desktop

Using semi-structured data for assessing research paper similarity
Germán Hurtado Martín, Steven Schockaert, Chris Cornelis, Helga Naessens (2013)
Information Sciences 221 p. 245-261
http://linkinghub.elsevier.com/retrieve/pii/S002002551200...
View in Mendeley Desktop

Figure 3.5: Mendeley's personal library page

The scope of Mendeley is therefore much broader than that of the previous systems and offers more and quite different features. Thanks to its desktop application, the reference manager can be integrated with text processors such as Microsoft Word or LaTeX, and articles in PDF form can be imported from other management programs to later annotate them. Also,

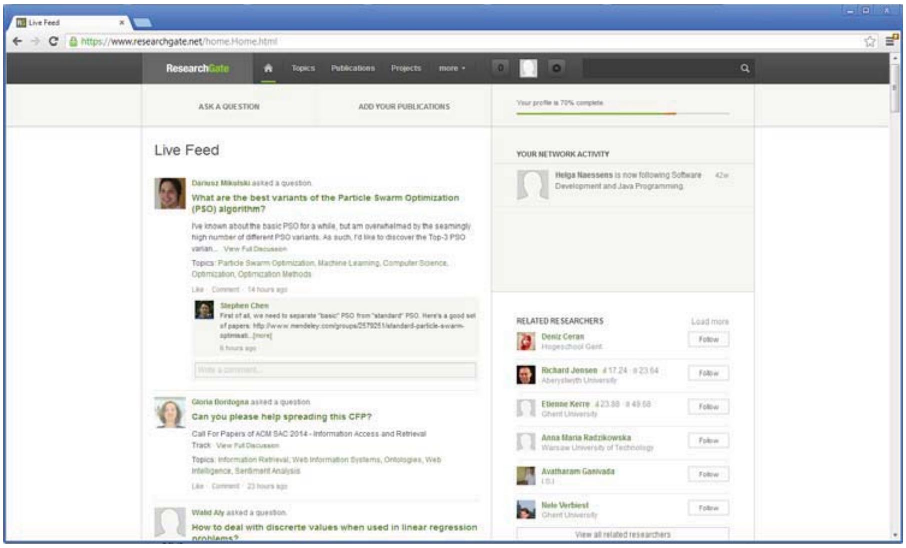
[26]http://www.mendeley.com

its social network makes it possible to create groups, share documents, follow other researchers activities, or discuss different topics.

What makes Mendeley worth discussing here are its recommendation features. On the one hand, when browsing a paper, the user gets the possibility to access "related research", which presents a list of similar papers. To this end, the system follows a content-based approach that uses several types of features: title, abstract, author, keyword, etc., as well as tags that can be added and edited by the users. This last feature has proven to be the most informative, although it has the important drawback that it is available only in about 20% of the cases [74]. On the other hand, users with an upgraded, paid-for version can get personalized recommendations via the Mendeley Suggest service. This service is based on collaborative filtering, more specifically item-based collaborative filtering [73].

3.3.6 ResearchGate

ResearchGate[27] is a social network for researchers. Launched in 2008, it could be described as something between CiteULike and the social network of Mendeley, although it is closer to the latter. The idea is to have a social networking site on which researchers can share data and publications, participate in discussions, follow other researchers, etc.



Dariusz Mikulski asked a question

What are the best variants of the Particle Swarm Optimization (PSO) algorithm?

I've known about the basic PSO for a while, but am overwhelmed by the seemingly high number of different PSO variants. As such, I'd like to discover the Top-3 PSO varian... View Full Discussion

Topics: Particle Swarm Optimization, Machine Learning, Computer Science, Optimization, Optimization Methods

Like · Comment · 14 hours ago

Stephen Chen
First of all, we need to separate "basic" PSO from "standard" PSO. Here's a good set of papers. http://www.mendeley.com/groups/2579251/standard-particle-swarm-optimisati...[more]
6 hours ago

Write a comment...

Gloria Bordogna asked a question

Can you please help spreading this CFP?

Call For Papers of ACM SAC 2014 - Information Access and Retrieval Track · View Full Discussion

Topics: Information Retrieval, Web Information Systems, Ontologies, Web Intelligence, Sentiment Analysis

Like · Comment · 23 hours ago

Walid Aly asked a question

How to deal with discrerte values when used in linear regression problems?

Helga Naessens is now following Software Development and Java Programming 42w

Deniz Ceran
Hogeschool Gent

Richard Jensen 17.24 · 23.64
Aberystwyth University

Etienne Kerre 23.88 · 49.68
Ghent University

Anna Maria Radzikowska
Warsaw University of Technology

Avatharam Ganivada
I.S.I

Nele Verbiest
Ghent University

Figure 3.6: ResearchGate's personal welcome page

Despite maintaining a profile for each user, each with his own library of research papers (most of them found by the system itself), ResearchGate

[27]http://www.researchgate.net

does not include a personal recommender. This is striking since, apart from content-based methods that could use the user's library, such a social networking site could also exploit the relations between users to use some kind of collaborative filtering approach.

ResearchGate does feature a "similar publications" function and for each paper three similar publications are presented. Again, the recommendation algorithm is undisclosed, and with so few recommendations it is difficult to conclude anything about the used method. However, it seems to be content-based, which sounds reasonable as the system offers the researchers the possibility of uploading the full text of the papers and that information could be used.

Chapter 4

Assessing research paper similarity

Research paper abstracts are usually accompanied by additional information such as keywords, authors, or journal. Our main goal in this chapter is to study to what extent this semi-structured information can be used to assess the similarity between two research papers following a content-based approach in the context of either the vector space model or language modeling. For the vector space model, we first consider the traditional tf-idf approach as a baseline method, and then investigate the potential of Explicit Semantic Analysis. In particular, we adapt a method from [46], 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 [154]. 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 discovered 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 chapter is structured as follows. In Section 4.1 we analyze the information usually available to compare two research papers. In Section 4.2 we study two methods based on the vector space model to measure article similarity, while in Section 4.3 we propose a number of methods based on language modeling. In Section 4.4 we explain the details concerning our experimental set-up, and in Section 4.5 we present and discuss the obtained results. The main conclusions are summarized in Section 4.7.

4.1 Available information

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, its authors, or its journal can be accessed. To illustrate this, we can take a look at the following paper: *"(v, T)-fuzzy rough approximation operators and the TL-fuzzy rough ideals on a ring"*. This is a 15-page paper, but unless one is subscribed to its publisher's services, the only available information is the one shown in Table 4.1. As we can see, a 15-page text is now reduced to a 4-sentence abstract. The amount of information thus becomes severely reduced.

Table 4.1: Information available about the considered paper

Title	(v, T)-fuzzy rough approximation operators and the TL-fuzzy rough ideals on a ring
Abstract	In this paper, we consider a ring as a universal set and study (v,T)-fuzzy rough approximation operators with respect to a TL-fuzzy ideal of a ring. First, some new properties of generalized (v, T)-fuzzy rough approximation operators are obtained. Then, a new fuzzy algebraic structure - TL-fuzzy rough ideal is defined and its properties investigated. And finally, the homomorphism of (v, T)-fuzzy rough approximation operators is studied.
Keywords	(v,T)-fuzzy rough approximation operator; TL-fuzzy ideal; (v,T)-fuzzy rough ideal; L-fuzzy relation; T-congruence
Authors	Fei Li; Yunqiang Yin; Lingxia Lu
Journal	Information Sciences
Year	2007
Bibliography	...

However, as stated above, the abstract is usually accompanied by document features that might be useful. Keywords are an example of such features, and intuitively they contain a considerable amount of information. They are supplied by the authors to give an idea about the concepts covered in the paper at a glance. This is also the reason why they have often been considered in recommenders, as mentioned in Chapter 3. In the example of Table 4.1, we see that the keywords indeed give a good idea about the topics covered in the paper. In this respect, it is interesting to note that when one reads these keywords one does not usually think only of these exact five concepts, but of slightly more general topics that also include synonyms, related terms, etc. We explore this idea in Section 4.3.3.

Another potentially useful feature are the authors' names. Most researchers usually write on the same topics, with a similar choice of words, so two papers written by the same person are likely to be related. Also, if we generalize that idea, we can think of scientific communities, instead of separate authors, where a community tends to cover the same concrete topic or group of topics. Again, two papers written by people within the same community are likely to be related. This information about the communities is not directly available, but can be discovered by means of LDA. We develop this idea in Section 4.3.3. An important limitation related to this feature is the problem of name ambiguity, as several authors may have the same name or one author may have several variations of his name. We further discuss this in Section 4.7.

The name of the journal may be useful as well, as the same journal usually covers the same topics. Therefore, intuitively, two randomly selected papers published in the same journal are more likely to be similar than two randomly selected papers published in different journals.

The publication year could be interesting to consider for different tasks, like paper recommendation, as a filtering feature (e.g. "I am only interested in papers published during the last six years"), or trend analysis, to see for example how the interest in some concepts has evolved in time. However, when the only goal is to assess the similarity between two papers this information is less useful, and therefore we do not consider it.

Also, the title is usually too short to consider on its own. It could be concatenated to the abstract in order to extend it; however, early results did not show significant improvement when the title was considered, probably due to the fact that most of the meaningful terms occurring in the title are usually already present in the abstract and/or the keywords. Therefore, we do not consider the title.

Finally, we might consider the bibliography. Papers citing the same works might be related (which is the idea behind the bibliographic coupling similarity measure [82]), as well as papers frequently cited together (the basis for co-citation similarity measure [129]), among other possibilities also based on the frequency and patterns of citations in papers. In practice, these techniques should be considered along with those proposed in the remainder of this chapter. However, they fall into citation analysis territory, which is out of the scope of this work as here we focus on the less studied content-based methods, and therefore we are not considering this feature.

Summing up, we will assess the similarity of research papers based on their abstract, keywords, authors, and journal. The challenge thus becomes to make optimal use of this limited amount of information.

4.2 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, and then another one based on Explicit Semantic Analysis [46].

4.2.1 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 w_i in the abstract of document d, the tf-idf scoring technique is used as defined in Chapter 2:

$$\mathit{tfidf}(w_i, d) = \frac{n(w_i, d)}{|d|} \cdot log\frac{|\mathcal{C}|}{|\{d_j : w_i \in d_j\}| + 1} \tag{4.1}$$

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, Dice, generalized Jaccard, and extended Jaccard similarity, defined respectively by Eqs. (2.7), (2.9), (2.10), and (2.11).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:

$$\mathit{tfidf}(w_i, d) = \frac{1}{|d|} \cdot log\frac{|\mathcal{C}|}{|\{d_j : w_i \in d_j\}| + 1} \tag{4.2}$$

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, here we consider the whole keywords, which may be multigrams (e.g. “recommender system”).

4.2.2 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

[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*.

in the *keywords* method which does not use the abstracts, intuitively the main source of information. 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 Analysis (ESA), as it is analogous to the approach from [46], which was discussed in Chapter 2. However, while the concepts in the original method come from Wikipedia, the concepts we use in this approach are features found in the papers. Specifically, if the chosen features are the keywords, 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 reflects how related the document is with the concept represented by the corresponding keyword.

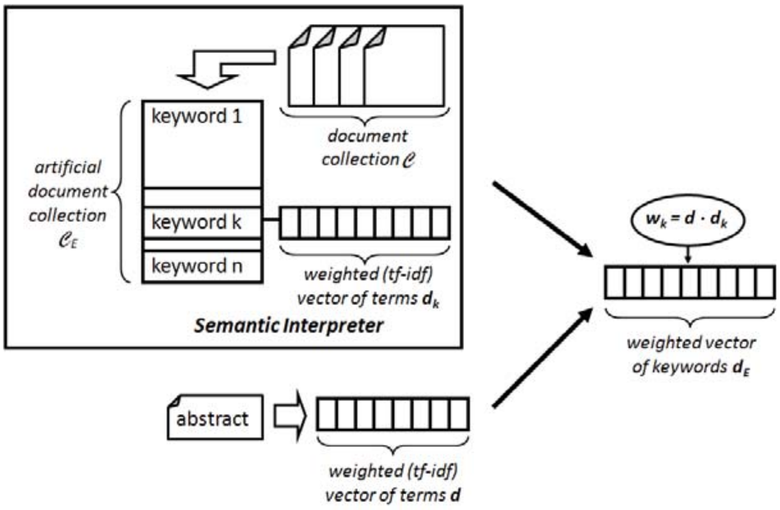

keyword 1
keyword k
keyword n
artificial document collection $\mathcal{C}_E$
document collection $\mathcal{C}$
weighted (tf-idf) vector of terms $\mathbf{d}_k$
Semantic Interpreter
$w_k = d \cdot d_k$
weighted vector of keywords $\mathbf{d}_E$
abstract
weighted (tf-idf) vector of terms $\mathbf{d}$

Figure 4.1: Keyword-based generation of the ESA vector $\mathbf{d_E}$ of a document

We now reformulate the explanation given in Chapter 2, according to the new source of information for the concepts. 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. This new collection contains a document d_k for each keyword, where d_k consists of the concatenation of the abstracts of the documents from the original collection $\mathcal{C}$ to which keyword k was assigned. Then, a weighted vector $\mathbf{d_k}$ is considered for each d_k. In this weighted vector, each component corresponds to a term in $\mathcal{C}_E$, and 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, $\mathbf{d}$ and $\mathbf{d_k}$ are normalized and can be compared to compute the new vector representation $\mathbf{d_E}$ of document d. In particular, the weight w_k in $\mathbf{d_E}$ of the component corresponding to keyword k is calculated as follows:

$$w_k = \mathbf{d} \cdot \mathbf{d_k} \tag{4.3}$$

Figure 4.1 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 2.1.4.

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 4 papers of the collection in the *ESA-aut* method, and only keywords that appear in at least 6 papers in the *ESA-kws* method.

4.3 Language modeling

4.3.1 Baseline

As an alternative to the approaches based on the vector space model, we consider language modeling, as language modeling techniques have already been shown to perform well for comparing short text snippets [62, 118]. As explained in Section 2.2, the idea underlying language modeling is that a document d is generated by a given probabilistic model D, where D has a probability for each word: the probability of that word being used to generate document d. Thus, what we want to do is to estimate unigram language models [117] for each document, and to evaluate their divergence. Each model is estimated from the terms that occur in the abstract of d (and the rest of the abstracts in the collection for the smoothing). Using Jelinek-Mercer smoothing, the probability that model D generates term w is given by:

$$D(w) = \lambda P(w|d) + (1 - \lambda)P(w|\mathcal{C}) \tag{4.4}$$

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 as defined in Section 2.2:

$$P(w|d) = \frac{n(w,d)}{|d|} \tag{4.5}$$

$$P(w|\mathcal{C}) = \frac{n(w,\mathcal{C})}{|\mathcal{C}|} \tag{4.6}$$

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:

$$KLD(D_1||D_2) = \sum_w D_1(w) log \frac{D_1(w)}{D_2(w)} \tag{4.7}$$

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

4.3.2 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, authors, and journal. In particular, the idea of interpolating language models, which underlies Jelinek-Mercer smoothing, can be generalized. Now we estimate and interpolate models also for the keywords k, authors a, and journal j of the paper:

$$D(w) = \lambda_1 P(w|d) + \lambda_2 P(w|k) + \lambda_3 P(w|a) + \lambda_4 P(w|j) + \lambda_5 P(w|\mathcal{C}) \tag{4.8}$$

with $\sum_i \lambda_i = 1$. Interpolation of language models has also been used for example in [154] 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 in which 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:

$$P(w|k) = \frac{1}{K} \sum_{i=1}^{K} P(w|k_i) \tag{4.9}$$

$$P(w|a) = \frac{1}{A} \sum_{j=1}^{A} P(w|a_j) \tag{4.10}$$

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 assign more importance to the first author by giving a higher weight γ to his probabilities. In that case, if there is more than one author, Eq. (4.10) becomes:

$$P(w|a) = \gamma P(w|a_1) + \frac{1-\gamma}{A-1} \sum_{j=2}^{A} P(w|a_j) \tag{4.11}$$

4.3.3 Latent Dirichlet Allocation

Two conceptually related abstracts may contain different terms (e.g. synonyms, misspellings, related terms), and may therefore not be recognized as being 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) [21], as explained in Chapter 2:

$$P(w|z) = \hat{\phi}_z^{(w)} = \frac{{n_z}^{(w)} + \beta}{{n_z}^{(\cdot)} + W\beta} \tag{4.12}$$

$$P(z|\tau) = \hat{\theta}_z^{(d)} = \frac{{n_z}^{(d)} + \alpha}{n_\cdot^{(d)} + T\alpha} \tag{4.13}$$

$$P(z|w,\tau) \propto P(w|z) \times P(z|\tau) = \frac{{n'_z}^{(w)} + \beta}{{n'_z}^{(\cdot)} + W\beta} \cdot \frac{{n'_z}^{(d)} + \alpha}{n'^{(d)}_\cdot + T\alpha} \tag{4.14}$$

Analogously to underlying topics, we can try to identify underlying scientific communities, as similar papers are often written by authors within the same community. 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 equations (4.12), (4.13) and (4.14) as follows

$$P(w|q) = \hat{\phi}_q^{(w)} = \frac{{n_q}^{(w)} + \beta}{{n_q}^{(\cdot)} + W\beta} \tag{4.15}$$

$$P(q|\kappa) = \hat{\theta}_q^{(d)} = \frac{{n_q}^{(d)} + \alpha}{n_\cdot^{(d)} + C\alpha} \tag{4.16}$$

$$P(q|w,\kappa) \propto P(w|q) \times P(q|\kappa) = \frac{{n'_q}^{(w)} + \beta}{{n'_q}^{(\cdot)} + W\beta} \cdot \frac{{n'_q}^{(d)} + \alpha}{n'^{(d)}_\cdot + C\alpha} \tag{4.17}$$

where C is the number of communities, q is a given community and κ is the new LDA model obtained with Gibbs sampling. The various counts are defined as described in Table 4.2. 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 4.5.3 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

Table 4.2: Values used in LDA with Gibbs sampling to find underlying communities

value	*description*
${n_q}^{(w)}$	Number of times term w is assumed to have been generated by community q.
${n_q}^{(d)}$	Number of times a term instance of document d is assumed to have been generated by community q.
${n_q}^{(\cdot)}$	Total number of times a term has supposedly been generated by community q.
$n_\cdot^{(d)}$	Total number of term instances of document d generated by any community.
${n'_q}^{(w)}$	Number of times term w is assumed to have been generated by community q, but without counting the current assignment of w.
${n'_q}^{(d)}$	Number of times a term instance of document d is assumed to have been generated by community q, but without counting the current assignment of w.
${n'_q}^{(\cdot)}$	Total number of times a term has supposedly been generated by community q, but without counting the current assignment of w.
${n'_\cdot}^{(d)}$	Total number of term instances of document d generated by any community, but without counting the current assignment of w.

propose to add $P(w|\tau)$ and $P(w|\kappa)$ to the right-hand side of Eq. (4.8), with the appropriate weights λ_i:

$$\begin{aligned} D(w) = \lambda_1 P(w|d) + \lambda_2 P(w|k) + \lambda_3 P(w|a) + \lambda_4 P(w|j) \\ + \lambda_5 P(w|\tau) + \lambda_6 P(w|\kappa) + \lambda_7 P(w|\mathcal{C}) \end{aligned} \tag{4.18}$$

$P(w|\tau)$ reflects the probability that term w is generated by the topics underlying document d. As defined in Chapter 2, it can be estimated by considering that:

$$P(w|\tau) = \sum_{i=1}^{T} P(w|z_i) \times P(z_i|\tau) \tag{4.19}$$

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

$$P(w|\kappa) = \sum_{i=1}^{C} P(w|q_i) \times P(q_i|\kappa) \tag{4.20}$$

In summary, we can now build a model D for each document d interpolating not only some features as in Eq. (4.8), but also underlying information such as topics and communities as defined in Eq. (4.18). This method is further referred to as *LM0*.

4.3.4 Enriched estimations

Equation (4.13) 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 probability that an article covers a given topic: authors usually write on the same topics, and journals cover a more or less well-defined spectrum. Therefore, we propose to use both features to compute the estimations of ϕ and θ.

While the outline of the method remains the same, we rewrite Eq. (4.13) as

$$P(z|\tau) = \hat{\theta}_z^{(d)} = \frac{n_z{}^{(d)} + \alpha_1 + n_z{}^{(j)}\alpha_2 + n_z{}^{(a)}\alpha_3}{n_\cdot{}^{(d)} + T\alpha_1 + n_\cdot{}^{(j)}\alpha_2 + n_\cdot{}^{(a)}\alpha_3} \tag{4.21}$$

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 Eq. (4.13) 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 4.5.3. This modification implies changes in the Gibbs sampling algorithm as well, replacing the part of the estimation of $P(z|\tau)$ in Eq. (4.14) by Eq. (4.21). We call this method *LM0e*.

4.3.5 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 (i.e., lines 1-12 of Algorithm 1 in Section 2.3), 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 4.2.2). Similarity between keywords is then estimated by calculating the cosine similarity between the corresponding artificial documents, and the clusters are obtained using the K-means algorithm [95]. We have chosen this basic clustering algorithm because it is fast, well known, and easy to implement.

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 4.5.3). 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 4.3.6, and in particular to Section 4.3.6.4.

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 Eqs. (4.12) and (4.13) have been set, we have two possibilities. On the one hand, we can work directly with these values, i.e., use them in Eqs. (4.12) and (4.13) or, in other words, proceed directly to line 31 of Algorithm 1. We call this method *LM1*. On the other hand, we can apply the iterative part of the LDA algorithm, i.e., start from line 13 of Algorithm 1. We refer to this method as *LM2*.

Both methods *LM1* and *LM2* can also be used to improve the community models, by following the same clustering and initialization process, but using authors instead of keywords. As with the topic models, the iterative part of the algorithm is skipped in *LM1*, and applied in *LM2*.

This idea can also be used to improve *LM0e*. In that case, the counts ${n_z}^{(j)}$, ${n.}^{(j)}$, ${n_z}^{(a)}$ and ${n.}^{(a)}$ 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 methods *LM1* and *LM2*; we call these methods *LM1e* and *LM2e* respectively.

4.3.6 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:

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

4.3.6.1 Step 1: basic language models

As explained in Section 4.3.1, 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:

$$\begin{array}{lll} P(a|d_1) = \frac{3}{6} & P(b|d_1) = \frac{1}{6} & P(c|d_1) = \frac{1}{6} \\ P(d|d_1) = \frac{1}{6} & P(e|d_1) = 0 & \end{array}$$

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

$$\begin{array}{lll} P(a|\mathcal{C}) = \frac{10}{19} & P(b|\mathcal{C}) = \frac{5}{19} & P(c|\mathcal{C}) = \frac{1}{19} \\ P(d|\mathcal{C}) = \frac{2}{19} & P(e|\mathcal{C}) = \frac{1}{19} & \end{array}$$

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

$$\begin{array}{ll} D_1(a) = \lambda \cdot \frac{3}{6} + (1-\lambda) \cdot \frac{10}{19} & D_1(b) = \lambda \cdot \frac{1}{6} + (1-\lambda) \cdot \frac{5}{19} \\ D_1(c) = \lambda \cdot \frac{1}{6} + (1-\lambda) \cdot \frac{1}{19} & D_1(d) = \lambda \cdot \frac{1}{6} + (1-\lambda) \cdot \frac{2}{19} \\ D_1(e) = \lambda \cdot 0 + (1-\lambda) \cdot \frac{1}{19} & \end{array}$$

4.3.6.2 Step 2: interpolated language models

However, as proposed in Section 4.3.2, 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:

$$\begin{array}{l} k_1 = \{a,b,a,c,d,a,a,a,d,a,b,a\} \\ k_2 = \{a,b,a,c,d,a,a,b,a\} \\ k_3 = \{a,a,d,a,b,a,a,b,a\} \end{array}$$

$k_4 = \{a, b, b, e\}$

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

$$\begin{array}{lll} P(a|k_1) = \frac{7}{12} & P(b|k_1) = \frac{2}{12} & P(c|k_1) = \frac{1}{12} \\ P(d|k_1) = \frac{2}{12} & P(e|k_1) = 0 & \end{array}$$

The same is done for k_2, k_3 and k_4. The same process is repeated for the authors and the journal: an artifical 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. (4.8). Some examples:

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

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.

4.3.6.3 Step 3: Latent Dirichlet Allocation

In Section 4.3.3 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 2 underlying topics, A and B. Then, as explained in Section 4.3.3, we need some counts to estimate the required probabilities defined in Eqs. (4.12) and (4.13). 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:

$$\begin{array}{lllll} {n_A}^{(a)} = 7 & {n_A}^{(b)} = 3 & {n_A}^{(c)} = 0 & {n_A}^{(d)} = 1 & {n_A}^{(e)} = 0 \\ {n_B}^{(a)} = 3 & {n_B}^{(b)} = 2 & {n_B}^{(c)} = 1 & {n_B}^{(d)} = 1 & {n_B}^{(e)} = 1 \end{array}$$

$$\begin{array}{llll} {n_A}^{(d_1)} = 4 & {n_A}^{(d_2)} = 3 & {n_A}^{(d_3)} = 3 & {n_A}^{(d_4)} = 1 \\ {n_B}^{(d_1)} = 2 & {n_B}^{(d_2)} = 3 & {n_B}^{(d_3)} = 0 & {n_B}^{(d_4)} = 3 \end{array}$$

${n_A}^{(\cdot)} = 11$, 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:

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

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 Eq. (4.19), 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 Eq. (4.18). For example:

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

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.

4.3.6.4 Step 4: LDA improvements

Section 4.3.4 and Section 4.3.5 propose how to improve the previously explained method. To enrich the estimations new variables are introduced in Eq. (4.21). In order to use these variables, we consider artificial documents as in Section 4.3.6.2, 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 4.3.5, 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:

$$\begin{aligned} &A = \{k_1, k_2\} \\ &B = \{k_3\} \end{aligned}$$

with their respective artificial documents c_A and c_B:

$$\begin{aligned} &c_A = \{a, b, a, c, d, a, a, a, d, a, b, a, a, b, a\} \\ &c_B = \{a, a, d, a, b, a, a, b, a\} \end{aligned}$$

According to the information in the clusters, topic A has generated term a 9 times, and B has generated a 6 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 this 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 these variables for example:

$$\begin{array}{lllll} {n_A}^{(a)} = 8 & {n_A}^{(b)} = 3 & {n_A}^{(c)} = 1 & {n_A}^{(d)} = 2 & {n_A}^{(e)} = 0 \\ {n_B}^{(a)} = 3 & {n_B}^{(b)} = 1 & {n_B}^{(c)} = 0 & {n_B}^{(d)} = 0 & {n_B}^{(e)} = 1 \end{array}$$

$$\begin{array}{llll} {n_A}^{(d_1)} = 5 & {n_A}^{(d_2)} = 4 & {n_A}^{(d_3)} = 3 & {n_A}^{(d_4)} = 2 \\ {n_B}^{(d_1)} = 1 & {n_B}^{(d_2)} = 2 & {n_B}^{(d_3)} = 0 & {n_B}^{(d_4)} = 2 \end{array}$$

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 4.3.6.3, 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 starts with more realistic/correct assumptions.

With both improvements, the rest of the LDA process remains the same. When the final models have been calculated, they can be compared by using Eq. (4.7).

4.4 Experimental set-up

To build a test collection and evaluate the proposed methods, we downloaded a portion of the ISI Web of Science[2], 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 25964 paper descriptions were retrieved, although our experiments are restricted to the 16597 papers for which none of the considered fields is empty.

The ground truth for our experiments is based on annotations made by 8 experts[3]. 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 **p** is thus compared against 30 others[4], some of which are tagged as similar. The approaches for assessing paper similarity discussed in Sections 4.2 and 4.3 can then be used to rank the 30 papers, such that ideally the papers similar to **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:

$$MAP = \frac{\sum_{r=1}^{|R|} AvPrec(r)}{|R|} \tag{4.22}$$

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

$$AvPrec = \frac{\sum_{i=1}^{n} Prec(i) \times rel(i)}{number\ of\ relevant\ documents} \tag{4.23}$$

  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).

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

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

[4]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.

- *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:

$$MRR = \frac{\sum_{r=1}^{|R|} RR(r)}{|R|} \tag{4.24}$$

  where RR is the reciprocal rank of a ranking:

$$RR = \frac{1}{fhit} \tag{4.25}$$

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

4.5 Experimental results

4.5.1 Vector space model

Table 4.3: Results obtained with the approaches based on the vector space model (methods described in Section 4.2)

MAP	*cos*	*dice*	*e.jacc*	*g.jacc*
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	*cos*	*dice*	*e.jacc*	*g.jacc*
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 4.3 summarizes the results of the experiment for the approaches based on the vector space model, as described in Section 4.2. In this table it is interesting to observe that the *abstract* method, traditionally combined with

cosine similarity, performs significantly better when instead combined with the general Jaccard similarity measure (paired t-test, $p < 0.001$). This is why we have built the ESA $\mathbf{d_E}$ vectors not only using the cosine similarity, as defined in Eq. (4.3) (results in the second block of the table) but also by replacing it by the generalized Jaccard similarity (results in the third block). In this last case there is an improvement when the resulting $\mathbf{d_E}$ vectors are compared using the cosine similarity, but when using any of the other three similarity measures the results are slightly worse. On the other hand, neither *ESA-kws* nor *ESA-aut* can outperform *abstract*, despite using two types of 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*, although that is probably related to the fact that all considered journals belong to the same domain.

4.5.2 Language modeling

Table 4.4 shows the results obtained with the language modeling methods, described in Section 4.3. The λ configurations in the first columns correspond to those controlling the weight of abstract, keywords, authors, journal, topics, and communities, in that order.

The first block of the table summarizes the results obtained with language models that only use one of these feature types. 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. 4.2 shows that giving a higher weight to the first author when modeling a paper, as proposed in Section 4.3.2, does not make a big difference.

Table 4.4: Results obtained with the approaches based on language modeling (methods described in Section 4.3)

	λ-configuration						MAP						MRR					
line	abs	kws	aut	jou	tpc	com	LM0	LM0e	LM1	LM1e	LM2	LM2e	LM0	LM0e	LM1	LM1e	LM2	LM2e
1	0.9	0	0	0	0	0	0.622	0.622	0.622	0.622	0.622	0.622	0.791	0.791	0.791	0.791	0.791	0.791
2	0	0.9	0	0	0	0	0.558	0.558	0.558	0.558	0.558	0.558	0.73	0.73	0.73	0.73	0.73	0.73
3	0	0	0.9	0	0	0	0.557	0.557	0.557	0.557	0.557	0.557	0.711	0.711	0.711	0.711	0.711	0.711
4	0	0	0	0.9	0	0	0.314	0.314	0.314	0.314	0.314	0.314	0.382	0.382	0.382	0.382	0.382	0.382
5	0	0	0	0	0.9	0	0.505	0.588	0.387	0.42	0.523	0.585	0.655	0.762	0.512	0.565	0.674	0.751
6	0	0	0	0	0	0.9	0.491	0.491	0.491	0.491	0.491	0.491	0.621	0.621	0.621	0.621	0.621	0.621
7	0.7	0	0	0	0.2	0	0.642	0.657	0.63	0.633	0.647	0.655	0.805	0.824	0.797	0.801	0.798	0.82
8	0.2	0	0	0	0.7	0	0.607	0.644	0.63	0.636	0.625	0.644	0.774	0.809	0.785	0.795	0.775	0.814
9	0.45	0	0	0	0.45	0	0.648	0.662	0.632	0.636	0.655	0.66	0.816	0.824	0.794	0.798	0.804	0.819
10	0.7	0.2	0	0	0	0	0.625	0.625	0.625	0.625	0.625	0.625	0.793	0.793	0.793	0.793	0.793	0.793
11	0.2	0.7	0	0	0	0	0.574	0.574	0.574	0.574	0.574	0.574	0.746	0.746	0.746	0.746	0.746	0.746
12	0.45	0.45	0	0	0	0	0.597	0.597	0.597	0.597	0.597	0.597	0.773	0.773	0.773	0.773	0.773	0.773
13	0.4	0.1	0	0	0.4	0	0.671	0.68	0.668	0.667	0.681	0.678	0.822	0.826	0.817	0.817	0.823	0.824
14	0.1	0.4	0	0	0.4	0	0.61	0.624	0.591	0.591	0.611	0.624	0.776	0.791	0.758	0.756	0.773	0.783
15	0.4	0.4	0	0	0.1	0	0.612	0.619	0.607	0.607	0.616	0.619	0.777	0.789	0.775	0.777	0.781	0.785
16	0.3	0.3	0	0	0.3	0	0.632	0.651	0.62	0.62	0.641	0.648	0.791	0.815	0.786	0.783	0.797	0.802
17	0.4	0	0.1	0	0.4	0	0.66	0.659	0.646	0.647	0.67	0.66	0.812	0.818	0.812	0.812	0.805	0.806
18	0.4	0	0	0.1	0.4	0	0.649	0.667	0.632	0.636	0.655	0.665	0.801	0.825	0.791	0.795	0.802	0.818
19	0.3	0.1	0.1	0.1	0.3	0	0.667	0.669	0.654	0.655	0.675	0.67	0.81	0.817	0.8	0.799	0.812	0.818
20	0.4	0.1	0.1	0	0.3	0	0.667	0.672	0.655	0.655	0.675	0.668	0.812	0.823	0.802	0.807	0.819	0.816
21	0.4	0.1	0	0.1	0.3	0	0.674	0.679	0.664	0.668	0.681	0.683	0.826	0.823	0.81	0.817	0.82	0.828
22	0.4	0	0	0	0.4	0.1	0.647	0.666	0.646	0.651	0.656	0.666	0.803	0.824	0.803	0.81	0.805	0.827
23	0.3	0.1	0	0	0.3	0.2	0.68	0.682	0.679	0.678	0.685	**0.687**	0.822	0.822	0.828	0.823	0.827	0.831
24	0.4	0.1	0.1	0	0.3	0.1	0.673	0.673	0.657	0.659	0.68	0.683	0.822	0.821	0.802	0.808	0.822	0.831
25	0.4	0.1	0	0.1	0.3	0.05	0.675	**0.687**	0.669	0.674	0.684	0.678	0.823	**0.834**	0.812	0.819	0.824	0.82

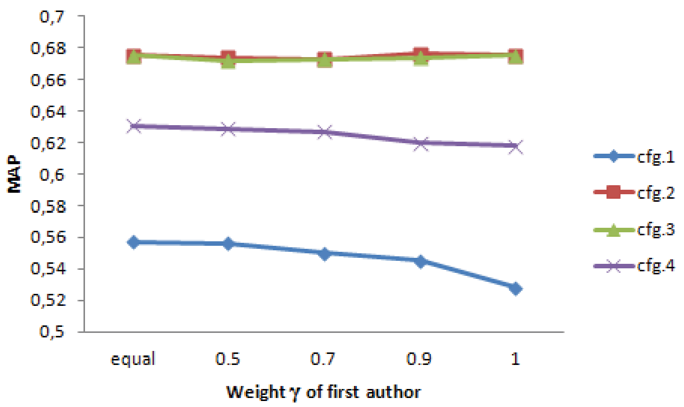


Figure 4.2: Impact of the first author's weight (configuration values shown in Table 4.5)

Table 4.5: 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

In the second block of Table 4.4 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*: $p \simeq 0.003$; *LM0e*: $p < 0.001$; *LM1*: $p \simeq 0.041$; *LM1e*: $p \simeq 0.024$; *LM2*: $p \simeq 0.001$; *LM2e*: $p < 0.001$). The differences are also significant between lines 8 and 11 for all methods except *LM0* (*LM0*: $p \simeq 0.062$; *LM0e*: $p < 0.001$; *LM1*: $p \simeq 0.003$; *LM1e*: $p \simeq 0.001$; *LM2*: $p \simeq 0.005$; *LM2e*: $p < 0.001$), and between lines 7 and 10 for *LM0e* and *LM2e* (*LM0e*: $p \simeq 0.022$; *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 all six methods *LM0*, *LM2* and *LM2e*). For all methods except *LM0e* and *LM2e*, the improvement is significant with respect to the configurations in lines 7-9 as well (*LM0*: $p < 0.029$; *LM1*: $p \simeq 0.002$; *LM1e*: $p \simeq 0.01$; *LM2*: $p < 0.012$); for *LM0e* the differences are only significant for lines 7-8 ($p \simeq 0.028$, resp. $p \simeq 0.001$; $p \simeq 0.055$ for line 9). Using authors and journal also means an improvement, but smaller than that achieved with the keywords. Combining more than three features, as in lines 19-21, does not show a significant improvement with respect to the previous lines. In Fig. 4.3 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 (especially for *LM1* and *LM1e*), but the optimal performance is obtained when both features are given approximately the same weight.

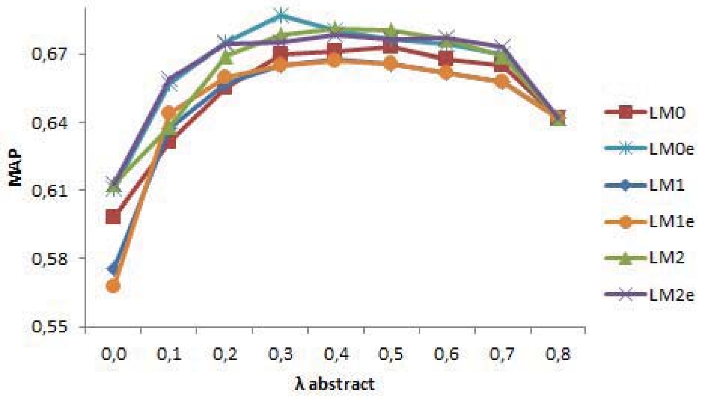


Figure 4.3: Importance of abstract vs. topics

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).

We can note that *LM1* and *LM1e* generally perform worse than *LM0*, and that *LM2* only slightly improves *LM0*. However, larger differences in MAP scores can be observed between *LM0* and methods *LM0e* 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 information of the different features.

4.5.3 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 Eq. (4.18)) 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 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 4.4 were obtained with 321 topics. Figures 4.4 and 4.5, however, show the limited importance of these choices with respect to the final results. Furthermore, parameters α and β introduced in Eqs. (2.20) and (2.21) 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 4.4) 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 4 documents and then we divided by 10. In Figure 4.6, however, we can observe the robustness of the method w.r.t. the choice of the value of C.

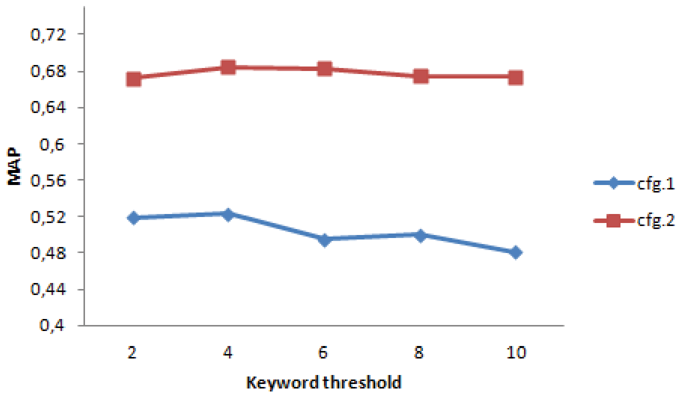


Figure 4.4: Impact of the keyword threshold, with $cfg.1 : \lambda_{tpc} = 0.9$ and $cfg.2 : \lambda_{abs} = 0.3, \lambda_{kws} = 0.1, \lambda_{tpc} = 0.3, \lambda_{com} = 0.2$.

As for methods *LM0e*, *LM1e* and *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 small weight. However, no weight is given to the journal, since preliminary ex-

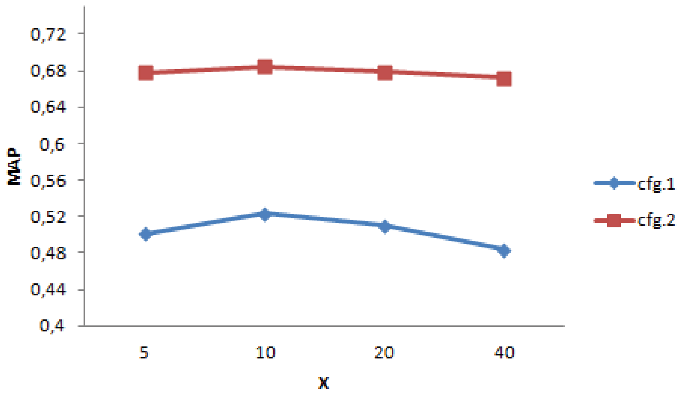


Figure 4.5: Impact of the number T of topics, $T = kws/X$, with $cfg.1$: $\lambda_{tpc} = 0.9$ and $cfg.2 : \lambda_{abs} = 0.3, \lambda_{kws} = 0.1, \lambda_{tpc} = 0.3, \lambda_{com} = 0.2$.

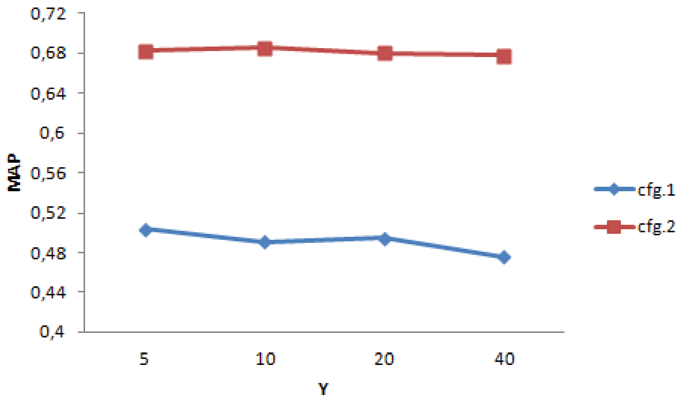


Figure 4.6: Importance of the number C of communities, $C = authors/Y$, with $cfg.1 : \lambda_{com} = 0.9$ and $cfg.2 : \lambda_{abs} = 0.3, \lambda_{kws} = 0.1, \lambda_{tpc} = 0.3, \lambda_{com} = 0.2$.

periments showed that the performance was not improved when using the journal information, as it was, like in *ESA-jou*, too general.

4.6 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 [117]. 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 [152]. A number of authors have investigated smoothing methods that go beyond the standard approaches. For instance, [89] 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. [38] 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. [80] and Petkova and Croft [115] try to improve smoothing by interpolating models, expanding the idea originally proposed by [77] on which we have partially built our approach. The idea of interpolating different language models was used in a particularly comprehensive way in [115]: 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 [103] evaluate, for the same task, models that combine author-based information with Dirichlet smoothing. Finally, [153] 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 [21]. Examples of this are the already mentioned methods of [103] and [153]. As discussed in Chapter 2, 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 [86], measuring the influence of Twitter users [145], or text classification [116]. 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 [18], 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 [24]; on the other hand [132] considers the intuition of authors usually writing about the same topics, and adds information about authors to create author-topic models, which in turn have been improved as well [48, 103]. 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 [20], or be associated with the topics in direct correspondence [120]. The approach of Kataria et al. [81] 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 4.3.4). However, our strategy uses no external sources of information, but only what is already in the document. Also, 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, which we have explored in methods LM1, LM1e, LM2 and LM2e (Section 4.3.5), appears to have received little attention in the literature.

4.7 Summary

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, 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 [128] which we have not addressed here. Also, 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 on concept representation (similarly to ESA), such as the one proposed in [47].

On the other hand, an 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.

Chapter 5

Content-based filtering of Calls For Papers

At the end of the previous chapter we mentioned the possibility of applying the methods we studied to a scientific paper recommender. However, while a number of techniques have been proposed recently for recommending scientific resources, with the study and emergence of research paper recommenders (see Chapter 3), citation recommendation [134], or applications to find experts in a specific research area [37], CFP recommendation remains unexplored to our knowledge. This is why we have decided to focus this chapter on a Call For Papers recommender rather than on a scientific paper recommender.

Nowadays many scientific conferences are organized, resulting in a high number of calls for papers (CFPs). This increasing number of CFPs, however, means for the researchers a substantial amount of time spent looking for potentially interesting conferences. The problem has been addressed in several ways, the most popular being the use of domain-specific mailing lists (e.g. DBWorld[1]), or organizing CFPs per subject on dedicated websites (e.g. WikiCFP[2], CFP List[3], or PapersInvited[4]). However, these solutions still require users to spend part of their time searching for CFPs, and the results do not always match their specific interests.

Recommenders typically rely on collaborative filtering approaches [135], content-based methods [113], or hybrid methods. It can be expected that a CFP recommender would be most effective when content-based methods are combined with other techniques. However, before such a recommender can be developed, we feel that a number of content-based aspects need to be understood better, including how the research interests of a user can

[1]http://research.cs.wisc.edu/dbworld/
[2]http://www.wikicfp.com
[3]http://www.cfplist.com
[4]http://www.papersinvited.com

be induced from his publication history and how these interests could be matched to CFPs. The aim of this chapter is to explore which methods may be most suitable for this task. In particular, we consider the textual content of the CFP such as the introductory text or the list of topics, and we complement that information with the abstracts of the papers recently written by the members of the program committee who are named in the CFP. On the other hand, we use information from the papers that the users have previously written to discover their research interests.

The chapter is structured as follows. First we discuss in more detail what types of information are at our disposal, and how this information can be used. Subsequently, in Section 5.2 we introduce different methods to effectively model and compare CFPs and user profiles. In Section 5.3 we describe the experiments performed and we show and discuss the results from these experiments. Finally, in Section 5.4 we summarize the conclusions of the chapter.

5.1 Available information

5.1.1 User representation

To represent the research interests of users we exploit the papers they have written. Since research interests might change, only recent papers are considered. In our experiments we have considered papers written in the last five years as being recent, although more advanced methods could be envisaged to analyze how the research interests of a user are changing over time. Alternatively, in the case of users with few or no papers (e.g. a beginning researcher) users could specify those papers which represent their interests best. As mentioned in Chapter 4, getting access to the full text of research papers is not always possible, and we therefore only use the papers' abstracts. We then consider, for each user, a document consisting of the concatenation of the abstracts of his papers. For the sake of clarity, we further refer to this document as d_{abs}.

What we can also learn from an author's publication profile is which authors he frequently cites. This information can be valuable if we consider that authors are more likely to be interested in conferences whose program committee (PC) contains several people who are working in the same field and whose papers they sometimes cite. To take this into account, we will use a second document consisting of the concatenation of the abstracts of the papers written by the authors usually cited by the user. In our experiments, we considered an author to be usually cited if at least 3 different papers written by him have been cited by the user in 3 different occasions. We refer to this document as d_{aut}.

5.1.2 CFP representation

For this work we have used CFPs available from DBWorld. Although there is no standard format for writing CFPs, they usually include similar information: an introductory text about the conference, an indicative list of topics that are within the scope of the conference, and the names of the members of the program committee (or at least the organizers). They usually also include important dates and location, but we will disregard that information.

The introductory text usually consists of a short description about the conference which might contain terms that describe the scope of the conference and are therefore important. However, this description often also refers to past conferences, the proceedings, etc., which means that many terms are mentioned that are not representative of the topics of the conference. We try to compensate this by concatenating the text of the CFP with the list of topics that are within the scope of the conference. We use the resulting document, which we further refer to as d_{txt}, to model a CFP document.

The names of the members of the program committee are also potentially useful. An option to use them directly could be trying to match them to the names cited in the papers of the users, but the results of initial experiments along these lines were not positive. However, these names can be used indirectly too. In particular, for the experiments reported in this paper, we associate each CFP with a document d_{con}, consisting of the concatenation of the abstracts of all papers that have been written in the last two years by the PC members.

Finally, if we want to consider both types of information simultaneously, we can concatenate d_{txt} and d_{con}; we refer to this document as d_{tot}. Table 5.1 summarizes the different types of information to represent users and CFPs.

Table 5.1: Different types of information for modeling users and CFPs

user	d_{abs}: concatenation of abstracts written by the users	d_{aut}: concatenation of abstracts written by frequently cited authors
CFP	d_{txt}: concatenation of introductory text and topics	d_{con}: concatention of abstracts written by the members of the PC
	d_{tot}: concatenation of d_{txt} and d_{con}	

5.2 Matching CFPs and users

Now that we have seen which kind of information is at our disposal, in this section we explore how to model it and how to compare the resulting models in order to assess the similarity between users and CFPs.

5.2.1 Tf-idf

To measure the similarity between a CFP and a user profile we compare them in the vector space model: each profile is represented as a vector, with one component for every term (unigram) occurring in the collection. A CFP is encoded as a vector analogously to Section 4.2.1: stopwords[5] are first removed, no stemming is used, and to calculate the weight for each term w_i in the CFP, the tf-idf scoring technique as defined in Eq. (2.4) is used. As mentioned in Section 5.1, CFPs can be represented in different ways. Depending on which concatenated document is used, the tf-idf score for each term is given by:

$$\mathit{tfidf}(w_i, d_{txt}) = \frac{n(w_i, d_{txt})}{|d_{txt}|} \cdot log(\frac{|\mathcal{C}_{txt}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.1}$$

$$\mathit{tfidf}(w_i, d_{con}) = \frac{n(w_i, d_{con})}{|d_{con}|} \cdot log(\frac{|\mathcal{C}_{con}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.2}$$

$$\mathit{tfidf}(w_i, d_{tot}) = \frac{n(w_i, d_{tot})}{|d_{tot}|} \cdot log(\frac{|\mathcal{C}_{tot}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.3}$$

where $\mathcal{C}_{txt}$ is the collection of CFPs made from the concatenation of introductory text and scope topics (i.e., of documents of the form d_{txt}), $\mathcal{C}_{con}$ is the collection of CFPs made from the concatenation of the abstracts of the papers written by the PC members (documents of the form d_{con}), and $\mathcal{C}_{tot}$ is the collection of CFPs made from the concatenation of both textual content and abstracts of the papers written by the PC members (documents of the form d_{tot}).

Since user profiles and CFPs belong to different collections, we consider user profiles as queries, and therefore the process to convert a user profile into a vector is slightly different. As with CFPs, stopwords are removed and no stemming is used; however, only those terms that occur in the CFP collection are considered, and the rest are ignored. Then the weight of each term in the user profile is calculated, depending on the type of information used:

$$\mathit{tfidf}(w_i, d_{abs}) = \frac{n(w_i, d_{abs}^{txt})}{|d_{abs}^{txt}|} \cdot log(\frac{|\mathcal{C}_{txt}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.4}$$

$$\mathit{tfidf}(w_i, d_{abs}) = \frac{n(w_i, d_{abs}^{con})}{|d_{abs}^{con}|} \cdot log(\frac{|\mathcal{C}_{con}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.5}$$

[5]The list of stopwords we have used is the same as in Section 4.2.1

$$\textit{tfidf}(w_i, d_{abs}) = \frac{n(w_i, d_{abs}^{tot})}{|d_{abs}^{tot}|} \cdot log(\frac{|\mathcal{C}_{tot}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.6}$$

where d_{abs}^{txt}, d_{abs}^{con} and d_{abs}^{tot} are obtained from the user profile d_{abs} after removing all terms that do not occur in $\mathcal{C}_{txt}$, $\mathcal{C}_{con}$ and $\mathcal{C}_{tot}$, respectively.

Two vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ corresponding to different profiles can then be compared using a standard similarity measure; we use the cosine similarity, as defined by Eq. (2.7), and the generalized Jaccard similarity, as defined by Eq. (2.10). Unlike in Chapter 4 we do not consider the extended Jaccard similarity nor the Dice similarity, since their performance was generally comparable or worse to that of the cosine similarity measure.

We further refer to the method that combines tf-idf with the cosine similarity measure as *tfidf-txt-cos*, *tfidf-con-cos* and *tfidf-tot-cos*, depending on the information used, and to the method that combines tf-idf with the generalized Jaccard similarity as *tfidf-txt-gja*, *tfidf-con-gja* and *tfidf-tot-gja*.

5.2.2 Language modeling

As in Chapter 4, we also consider the alternative of estimating unigram language models for each document, and determining their divergence. A user or CFP d is then assumed to be generated by a given model D. This model is estimated from the terms that occur in d and in the other CFPs. Using Jelinek-Mercer smoothing as in Eq. (2.15), the probability that model D corresponding to a CFP generates term w is estimated as:

$$P^*(w|D) = \lambda P(w|d_{txt}) + (1-\lambda)P(w|\mathcal{C}_{txt}) \tag{5.7}$$

$$P^*(w|D) = \lambda P(w|d_{con}) + (1-\lambda)P(w|\mathcal{C}_{con}) \tag{5.8}$$

depending on the type of information used, where $\mathcal{C}_{txt}$ and $\mathcal{C}_{con}$ are the collections of CFPs as defined in Section 5.2.1. The probabilities $P(w|d)$ and $P(w|\mathcal{C})$ are estimated using maximum likelihood, as defined in Eqs. (2.12) and (2.14) respectively. Again, stopwords are removed from d before estimating its model.

Alternatively, Dirichlet smoothing, as in (2.16), can be used:

$$P^*(w|D) = \frac{n(w, d_{txt}) + \mu P(w|\mathcal{C}_{txt})}{|d_{txt}| + \mu} \tag{5.9}$$

$$P^*(w|D) = \frac{n(w, d_{con}) + \mu P(w|\mathcal{C}_{con})}{|d_{con}| + \mu} \tag{5.10}$$

where $\mu = |d_{txt}| + 1$ and $\mu = |d_{con}| + 1$ respectively. Note that we do not use the average document length for μ as in Chapter 2, but the document length, which is not an unusual alternative [10] either. We add 1 for the cases where d_{con} is empty[6] (i.e., when the CFP has no PC and therefore no concatenation of abstracts written by its members).

To estimate the probability that the model of a user profile generates a given term w we simply replace d_{txt} in Eqs. (5.7) and (5.9) and d_{con} in Eqs. (5.8) and (5.10) by d_{abs}^{txt} and d_{abs}^{con} (as defined in Section 5.2.1) respectively.

Once the models D_1 and D_2 corresponding to a user profile d_1 and a CFP d_2 are estimated, we measure their dissimilarity using the Kullback-Leibler divergence:

$$KLD(D_1||D_2) = \sum_w D_1(w) log \frac{D_1(w)}{D_2(w)} \tag{5.11}$$

In Section 5.3.2, we refer to these methods as *lm-txt-jms* and *lm-con-jsm*, when Jelinek-Mercer smoothing is used, and as *lm-txt-dir* and *lm-con-dir* when Dirichlet smoothing is used.

However, if we want to consider both kinds of information jointly (i.e., the information from the documents of the form d_{txt} and that from the documents of the form d_{con}), language model interpolation is used. The idea of interpolating language models, which underlies Jelinek-Mercer smoothing, can be generalized:

$$P^*(w|D) = \lambda_1 P(w|d_{txt}) + \lambda_2 P(w|d_{con}) + \lambda_3 P(w|\mathcal{C}_{txt}) + \lambda_4 P(w|\mathcal{C}_{con}) \tag{5.12}$$

for the CFPs, and

$$P^*(w|D) = \lambda_1 P(w|d_{abs}^{txt}) + \lambda_2 P(w|d_{abs}^{con}) + \lambda_3 P(w|\mathcal{C}_{txt}) + \lambda_4 P(w|\mathcal{C}_{con}) \tag{5.13}$$

for the user profiles, with $\sum_i \lambda_i = 1$ and where

$$\lambda_3 = \begin{cases} \frac{1-\lambda_1-\lambda_2}{2}, & \text{if } \lambda_1, \lambda_2 > 0 \\ 1-\lambda_1, & \text{if } \lambda_2 = 0 \\ 0, & \text{if } \lambda_1 = 0 \end{cases} \qquad \lambda_4 = \begin{cases} \frac{1-\lambda_1-\lambda_2}{2}, & \text{if } \lambda_1, \lambda_2 > 0 \\ 0, & \text{if } \lambda_2 = 0 \\ 1-\lambda_2, & \text{if } \lambda_1 = 0 \end{cases}$$

In Section 5.3.2, we refer to this method as *lm-tot-jms*.

On the other hand, if Dirichlet smoothing is used we interpolate the models obtained with Eqs. (5.9) and (5.10):

$$P^*(w|D) = \lambda_1 (\frac{n(w, d_{txt}) + \mu P(w|\mathcal{C}_{txt})}{|d_{txt}| + \mu}) + \lambda_2 (\frac{n(w, d_{con}) + \mu P(w|\mathcal{C}_{con})}{|d_{con}| + \mu}) \tag{5.14}$$

for the CFPs, and

$$P^*(w|D) = \lambda_1 (\frac{n(w, d_{abs}^{txt}) + \mu P(w|\mathcal{C}_{txt})}{|d_{abs}^{txt}| + \mu}) + \lambda_2 (\frac{n(w, d_{abs}^{con}) + \mu P(w|\mathcal{C}_{con})}{|d_{abs}^{con}| + \mu}) \tag{5.15}$$

for the users profiles. We refer to this method as *lm-tot-dir*. Again, $\mu = |d| + 1$, where d is d_{txt}, d_{con}, d_{abs}^{txt} or d_{abs}^{con}, depending on the case.

[6]While d_{txt} cannot be empty, we set $\mu = |d_{txt}| + 1$ rather than $\mu = |d_{txt}|$ to keep things simple, i.e., μ is equal to the document length plus 1 in all cases.

5.2.3 Feature selection

As mentioned in Section 5.1, the introductory texts of the CFPs often contain information about past editions of the conference or brief submission guidelines. This leads to the use of a number of relatively common terms, which are irrelevant for characterizing the scope of a conference. To eliminate such unwanted terms, we use the term strength method described in Section 2.1.1.3. We recall that the *strength* of a term w is computed by estimating the probability that a term w occurs in a document d_1 given that it occurs in a related document d_2:

$$strength(w) = P(w \in d_1 | w \in d_2) \tag{5.16}$$

In this case, in order to construct the pairs of related documents we use method *tfidf-txt-cos* from Section 5.2.1. Also, as in Section 2.1.1.3, we set the threshold of the average number related documents per document (i.e., the average number of pairs (d_i, d_j) for each d_i) to a value between 10 and 20.

After calculating $strength(w)$ for every term w in the CFP collection, the N strongest terms are selected, ignoring the rest. For our experiments in Section 5.3 we have used $N = 500$ and $\mathcal{C}_{txt}$ as the CFP collection, since that combination performed well in early tests. The documents are then modelled as in Sections 5.2.1 and 5.2.2. When referring to particular methods in Section 5.3.2, we indicate when feature selection was used by adding the suffix *-fs* to the name of the method.

5.2.4 Related authors

As mentioned in Section 5.1, to reflect users' interest for those conferences whose PC members they are familiar with we propose to calculate extra models exclusively based on papers and compare them. Specifically, we compare the CFP model based on the concatenation of the abstracts of the papers written by the PC members (d_{con}) with a user model based on the concatenation of the abstracts of the papers written by the researchers usually cited by that particular user (d_{aut}). Depending on the used method, the model based on d_{con} is constructed according to Eq. (5.2), Eq. (5.8), or Eq. (5.10). For the model based on d_{aut} these definitions become

$$tfidf(w_i, d_{aut}) = \frac{n(w_i, d_{aut}^{con})}{|d_{aut}^{con}|} \cdot log(\frac{|\mathcal{C}_{con}|}{|\{d_j : w_i \in d_j\}|}) \tag{5.17}$$

$$P^*(w|D) = \lambda P(w|d_{aut}^{con}) + (1-\lambda)P(w|\mathcal{C}_{con}) \tag{5.18}$$

$$P^*(w|D) = \frac{n(w, d_{aut}^{con}) + \mu P(w|\mathcal{C}_{con})}{|d_{aut}^{con}| + \mu} \tag{5.19}$$

where $\mu = |d_{aut}^{con}| + 1$.

The method used to create and compare these extra models is always analogous to that used to calculate the original result, e.g. if the original result is obtained with method *lm-txt-jms* (language modeling with Jelinek-Mercer smoothing), Eqs. (5.8) and (5.18) are used to calculate these extra models, and they are then compared using the Kullback-Leibler divergence.

The idea is to use these models to complement the result obtained with the methods seen in the previous sections. In particular, once the models are created and compared, we simply combine the result with that of the original comparison by means of the weighted average. For example, to compare CFP *cfp* and user *u* with method *tfidf-txt-cos*, the result was given by $sim_c(\mathbf{cfp_{txt}}, \mathbf{u_{txt}})$. However, if we take into account these extra models based on d_{con} and d_{aut} (in this case, cfp_{con} and u_{aut}), the result is now given by:

$$\alpha \cdot sim_c(\mathbf{cfp_{txt}}, \mathbf{u_{txt}}) + \beta \cdot sim_c(\mathbf{cfp_{con}}, \mathbf{u_{aut}}) \tag{5.20}$$

where $\alpha + \beta = 1$. Based on preliminary experiments, we use $\alpha = 0.8$ and $\beta = 0.2$ for the experiments in Section 5.3.2. We indicate that these extra models are used by adding the suffix *-nam* to the name of the method.

5.2.5 Related authors & feature selection

Finally, both previously introduced variations can be combined: first, feature selection is applied, which also reduces the number of terms in the extra models based on the frequently cited authors, and then, as explained in the previous subsection, the models are compared separately, to finally combine the results. We indicate that this variation is used by adding the suffix *-fsn* to the name of the method.

5.3 Experimental evaluation

5.3.1 Experimental set-up

To build a test collection and evaluate the proposed methods, we downloaded 1769 CFPs posted between February and July 2012 at DBWorld, which reduced to 1152 CFPs after removing duplicates. Additionally, those CFPs lacking an introductory text or an indicative list of topics were removed too, which further reduced the total number to 969 CFPs. Each of these CFPs has a text part (concatenation of introductory text and topics) and a concatenation of the abstracts of the papers written by the PC members in the last 2 years[7], where available.

On the other hand, 13 researchers from a field which relates to the scope of DBWorld took part in our experiments as users. In order to profile them,

[7]All the information regarding research papers was retrieved from the ISI Web of Science, http://apps.isiknowledge.com .

we downloaded the abstracts of the papers they wrote in the last 5 years. The ground truth for our experiments is based on annotations made by these 13 users[8]. In a first experiment, each user indicated, for a minimum of 100 CFPs, whether these were relevant or not (relevance degree of 1 or 0 respectively). Then, using each of the studied methods, the CFPs annotated by the users were ranked such that ideally the relevant CFPs appear at the top of the ranking.

In a second experiment, we considered only CFPs assessed as highly relevant by at least one of the methods. To this end, we selected for each user and each of the 48 studied methods the top-5 CFPs of the rankings obtained in the first experiment. This resulted in 240 CFPs, which reduced to an average of about 50 CFPs per user due to overlap between the top-5 CFPs returned by each method. Each of those CFPs was then rated by the user, who gave them a score between 0 ("totally irrelevant") and 4 ("totally relevant"). Again, using each of the studied methods, these CFPs were ranked such that ideally the most relevant CFPs appear at the top of the ranking.

To evaluate the rankings resulting from both experiments, for each user and each method we use normalized discounted cumulative gain (nDCG) [76] to measure the relevance of each CFP according to its position in the ranking. The idea of this measure is that the greater the ranked position of a relevant document, the less valuable it is for the user, as users tend to examine only those documents ranked high, except if those documents do not satisfy their information needs, in which case it is more likely that they still consider lower ranked documents. This is reflected by the discounted cumulative gain of the document ranked in position r:

$$DCG_r = rel_1 + \sum_{i=2}^{r} \frac{rel_i}{log_2 i} \tag{5.21}$$

The relevance rel_i of the document ranked in position i is the relevance indicated by the user, i.e., 0 or 1 for the first experiment, and 0, 1, 2, 3 or 4 for the second experiment.

Since the number of CFPs annotated by each user might be different, the length of the obtained rankings varies. In order to compare the DCG values we need to calculate the normalized DCG:

$$nDCG_r = \frac{DCG_r}{iDCG_r} \tag{5.22}$$

where $iDCG_r$ is the ideal DCG at position r: the DCG obtained at position r in the ideal case where all documents are perfectly ranked, from most to least relevant, according to the users' annotations.

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

For both experiments in Section 5.3.2 we work with the nDCG of the CFP ranked in the last position, i.e., nDCG_r where r is the total number of CFPs in the ranking, as this value reflects the gains of all the CFPs throughout the whole ranking.

5.3.2 Results

Tables 5.2 and 5.3 summarize the results of the first and second experiment respectively. In particular, for each method we show the average nDCG_r for the 13 users, where r is the number of CFPs in the ranking for each user as indicated in the previous section. For the sake of simplicity we have used some fixed values for the λ parameters of (5.12) and (5.13) in the methods based on language modeling with Jelinek-Mercer smoothing. In particular, we use $\lambda_1 = 0.9$ and $\lambda_2 = 0$ for the *lm-txt-jms* method (i.e., analogously to the *tfidf-txt* methods, it only uses the information from the text parts of the CFPs); $\lambda_1 = 0$ and $\lambda_2 = 0.9$ for the *lm-con-jms* method; and $\lambda_1 = 0.4$ and $\lambda_2 = 0.4$ for the *lm-tot-jms* method. For method *lm-tot-dir*, that uses (5.14) and (5.15), we fix $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$.

First we compare the different kinds of information that can be used: introductory text plus topics (*txt*), concatenation of the abstracts of the papers recently written by the PC members (*con*), or the concatenation of both (*tot*). Figures 5.1 and 5.2 show that, in general, using the abstracts alone (d_{con}) does not suffice to outperform the methods based on the textual content (d_{txt}), except for methods *tfidf-con-cos*, *tfidf-con-gja*, and *tfidf-con-gja-nam* in Experiment 1. Actually, in most cases in both experiments, using the abstracts alone performs worse than using the textual content. However, only in a few cases are these differences significant[9], as shown in line 1 of Table 5.4 and Table 5.5. On the other hand, methods based on the concatenation of abstracts and textual content seem to perform comparably or slightly better than the *txt* methods, although in some cases the performance gets notably worse (*lm-tot-jms-nam* in Experiment 1; *tfidf-tot-cos*, *tfidf-tot-cos-nam*, *tfidf-tot-gja* and its variants, and *lm-tot-jms-nam* in Experiment 2). As shown in line 2 of Table 5.4 and Table 5.5, these differences are significant mainly in Experiment 2, but only for some methods. However, we can see that *tot* methods do significantly outperform *con* methods in most cases for both experiments, in particular for the methods based on language modeling.

To study the impact of feature selection (*fs*), the additional models based on frequently cited authors (*nam*) and the combination of both (*fsn*), we fix the method and the type of information used. As shown in Figs. 5.3 and 5.4, in general, the best results are obtained when feature selection is applied, especially for the methods based on language modeling. It must be

[9]In this chapter we consider a difference to be significant when $p < 0.05$ for the Mann-Whitney U test. It should be noted that the low number of users affects the significance.

Table 5.2: Ranking of methods for the first experiment, nDCG values

Method	nDCG	Method	nDCG	Method	nDCG
tfidf-tot-cos-fs	0.606	tfidf-con-gja-nam	0.548	lm-txt-jms-fsn	0.512
tfidf-tot-cos-fsn	0.599	tfidf-txt-gja-fsn	0.547	lm-txt-jms	0.51
lm-tot-dir-fs	0.581	tfidf-tot-gja-fsn	0.546	lm-tot-dir-nam	0.497
lm-tot-dir-fsn	0.578	tfidf-con-cos	0.544	lm-tot-jms	0.493
lm-tot-jms-fs	0.575	tfidf-tot-gja-fs	0.544	lm-con-jms-fs	0.493
tfidf-tot-gja	0.565	tfidf-txt-cos	0.542	lm-txt-jms-nam	0.482
tfidf-tot-cos	0.563	tfidf-txt-gja-fs	0.537	lm-txt-dir	0.471
tfidf-txt-cos-fsn	0.563	tfidf-txt-gja-nam	0.535	lm-txt-dir-nam	0.47
tfidf-txt-cos-nam	0.562	tfidf-con-gja-fsn	0.535	lm-con-jms-fsn	0.469
tfidf-tot-cos-nam	0.561	tfidf-con-gja-fs	0.532	lm-con-jms	0.44
tfidf-tot-gja-nam	0.556	lm-tot-dir	0.529	lm-tot-jms-nam	0.436
tfidf-txt-cos-fs	0.555	lm-txt-jms-fs	0.529	lm-con-jms-nam	0.421
tfidf-con-cos-fsn	0.553	lm-txt-dir-fsn	0.526	lm-con-dir-nam	0.418
tfidf-con-gja	0.552	tfidf-txt-gja	0.518	lm-con-dir-fs	0.415
tfidf-con-cos-nam	0.551	lm-tot-jms-fsn	0.516	lm-con-dir	0.414
tfidf-con-cos-fs	0.549	lm-txt-dir-fs	0.514	lm-con-dir-fsn	0.408

Table 5.3: Ranking of methods for the second experiment, nDCG values

Method	nDCG	Method	nDCG	Method	nDCG
lm-tot-jms-fs	0.745	lm-txt-jms-fsn	0.682	tfidf-con-cos-nam	0.636
tfidf-txt-cos-nam	0.728	tfidf-tot-cos	0.661	lm-tot-dir	0.612
tfidf-txt-cos	0.715	tfidf-tot-gja	0.657	lm-con-jms-fs	0.606
tfidf-tot-cos-fs	0.713	tfidf-tot-gja-fsn	0.655	lm-txt-dir-fsn	0.603
tfidf-txt-gja-fs	0.708	lm-tot-jms	0.653	lm-txt-dir-fs	0.6
tfidf-txt-cos-fsn	0.707	tfidf-tot-gja-fs	0.653	lm-con-jms-fsn	0.587
tfidf-tot-cos-fsn	0.706	tfidf-con-cos-fs	0.649	lm-con-dir-fs	0.569
tfidf-txt-cos-fs	0.705	tfidf-tot-cos-nam	0.648	lm-tot-jms-nam	0.566
lm-txt-jms-fs	0.700	tfidf-tot-gja-nam	0.648	lm-tot-dir-nam	0.562
tfidf-txt-gja-fsn	0.699	lm-txt-jms-nam	0.647	lm-txt-dir	0.559
lm-tot-dir-fs	0.693	tfidf-con-cos-fsn	0.646	lm-con-jms	0.555
lm-txt-jms	0.691	tfidf-con-gja-fs	0.645	lm-txt-dir-nam	0.545
tfidf-txt-gja	0.69	tfidf-con-gja	0.644	lm-con-dir	0.525
lm-tot-dir-fsn	0.687	tfidf-con-gja-nam	0.644	lm-con-dir-fsn	0.505
lm-tot-jms-fsn	0.686	tfidf-con-gja-fsn	0.642	lm-con-jms-nam	0.502
tfidf-txt-gja-nam	0.682	tfidf-con-cos	0.637	lm-con-dir-nam	0.493

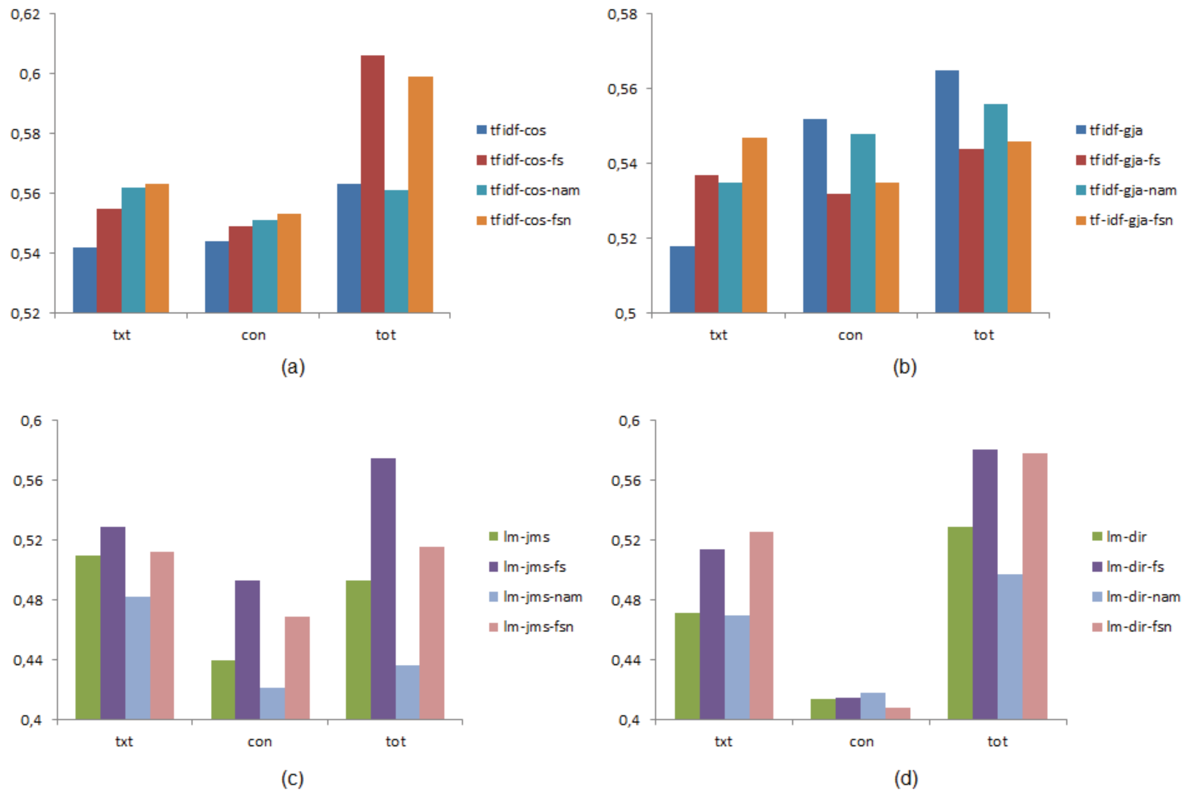


Figure 5.1: Results for experiment 1 with a) tf-idf combined with cosine similarity; b) tf-idf combined with generalized Jaccard similarity; c) language modeling with Jelinek-Mercer smoothing; d) language modeling with Dirichlet smoothing. The Y-axis shows the nDCG, while the X-axis indicates the kind of information used.

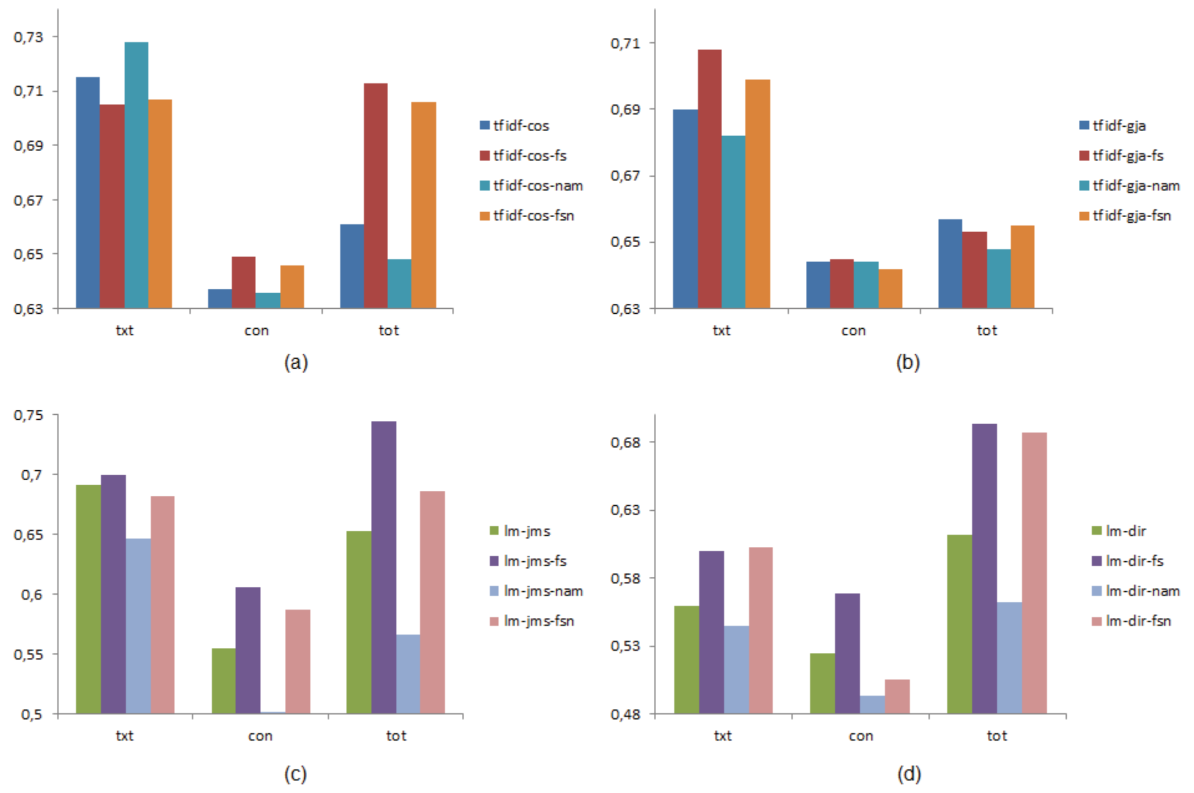


Figure 5.2: Results for experiment 2 with a) tf-idf combined with cosine similarity; b) tf-idf combined with generalized Jaccard similarity; c) language modeling with Jelinek-Mercer smoothing; d) language modeling with Dirichlet smoothing. The Y-axis shows the nDCG, while the X-axis indicates the kind of information used.

Table 5.4: Summary of significant differences based on the type of information used: *txt*, *con* and *tot*, compared as indicated by the second column; significant difference is indicated by +/- when the first method in the pair performs better/worse than the second one. Experiment 1

		cos				gja				jms				dir			
line	comp.	-	fs	nam	fsn	-	fs	nam	fsn	-	fs	nam	fsn	-	fs	nam	fsn
1	*txt - con*														+		
2	*txt - tot*														-		-
3	*tot - con*			+		+		+		+	+			+	+	+	+

Table 5.5: Summary of significant differences based on the type of information used: *txt*, *con* and *tot*, compared as indicated by the second column; significant difference is indicated by +/- when the first method in the pair performs better/worse than the second one. Experiment 2

		cos				gja				jms				dir			
line	comp.	-	fs	nam	fsn	-	fs	nam	fsn	-	fs	nam	fsn	-	fs	nam	fsn
1	*txt - con*			+						+		+				+	
2	*txt - tot*			+							-	+			-		-
3	*tot - con*	+		+		+				+	+	+	+	+	+	+	+

noted, however, that these differences are only significant in some cases, as depicted by line 1 of Tables 5.6 and 5.7: *lm-con-jms-fs* and *lm-tot-jms-fs* in Experiment 1, and *lm-tot-jms-fs* and *lm-tot-dir-fs* in Experiment 2. On the other hand, results obtained with the *nam* methods are worse than the original, with significant differences for *lm-jms* methods, mainly in Experiment 2. Finally, *fsn* usually improves the original results, but as shown in line 3 of Tables 5.6 and 5.7 there is no significant evidence of this.

Table 5.6: Summary of significant differences based on the variation used: none (-), *fs*, *nam* and *fsn*, compared as indicated by the second column; significant difference is indicated by +/- when the first method in the pair performs better/worse than the second one. Experiment 1

		cos			gja			jms			dir		
line	comp.	txt	con	tot	txt	con	tot	txt	con	tot	txt	con	tot
1	*no - fs*								-	-			
2	*no - nam*									+			
3	*no - fsn*												
4	*fs - nam*									+			
5	*fs - fsn*												
6	*nam - fsn*												

Table 5.7: Summary of significant differences based on the variation used: none (-), *fs*, *nam* and *fsn*, compared as indicated by the second column; significant difference is indicated by +/- when the first method in the pair performs better/worse than the second one. Experiment 2

		cos			gja			jms			dir		
line	comp.	txt	con	tot	txt	con	tot	txt	con	tot	txt	con	tot
1	*no - fs*									-			-
2	*no - nam*							+	+	+			
3	*no - fsn*												-
4	*fs - nam*						+	+	+	+		+	+
5	*fs - fsn*											+	
6	*nam - fsn*							-	-	-			-

If the variations are compared to each other (lines 4-6 of Tables 5.6 and 5.7) we see almost no significant differences in Experiment 1. In Experiment 2, however, it can be seen that *fs* significantly outperforms *nam*, especially for the methods based on language modeling, as does *fsn*. If we compare *fs* and *fsn*, although the results show some improvement of *fs* over *fsn*, the difference is in general not significant.

Table 5.9: Summary of significant differences based on the method used: *cos*, *gja*, *jms* and *dir*, compared as indicated by the second column; significant difference is indicated by +/- when the first method in the pair performs better/worse than the second one. Experiment 2

		txt				con				tot			
li	method	-	fs	nam	fsn	-	fs	nam	fsn	-	fs	nam	fsn
1	*cos - gja*			+									
2	*cos - jms*			+		+		+	+			+	
3	*cos - dir*	+	+	+	+	+		+	+			+	
4	*gja - jms*					+	+	+	+			+	
5	*gja - dir*	+	+	+	+	+		+	+			+	
6	*jms - dir*	+	+		+						+		

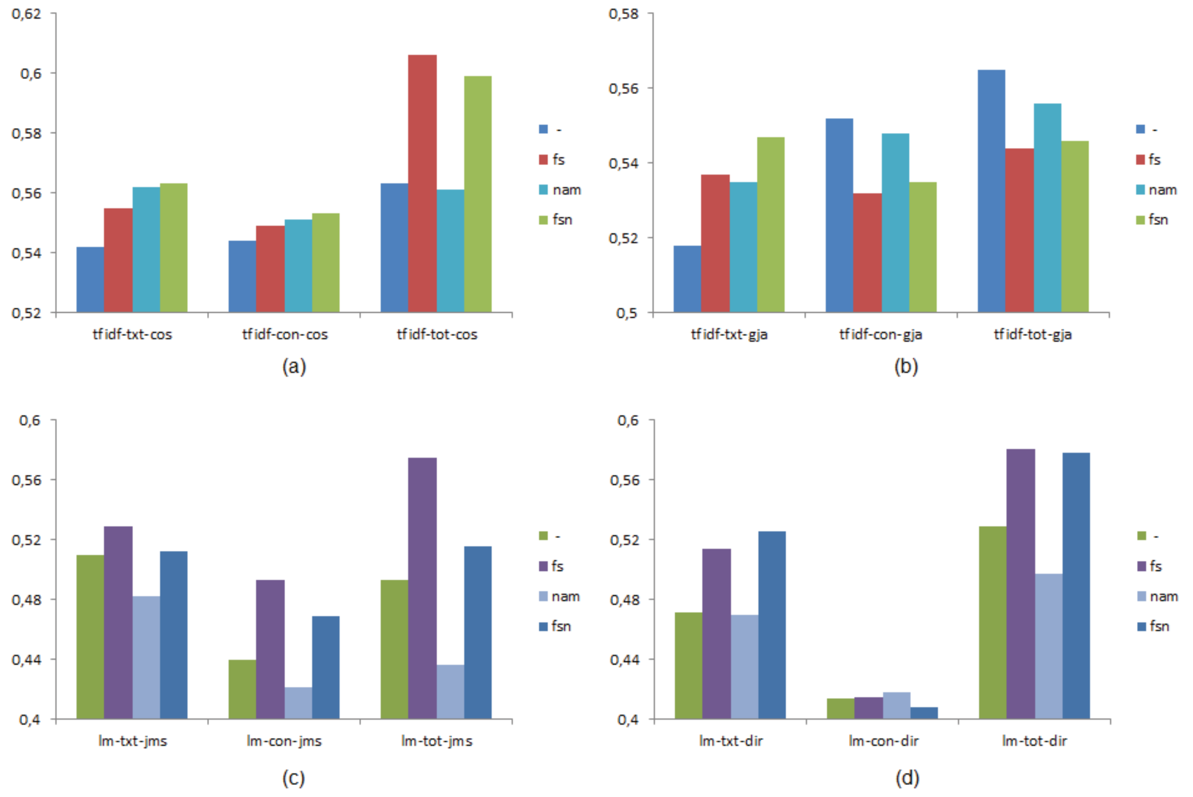


Figure 5.3: Comparison of variations (no variation, *fs*, *nam* or *fsn*) for experiment 1 with a) tf-idf combined with cosine similarity; b) tf-idf combined with generalized Jaccard similarity; c) language modeling with Jelinek-Mercer smoothing; d) language modeling with Dirichlet smoothing. The Y-axis shows the nDCG, while the X-axis indicates the method used.

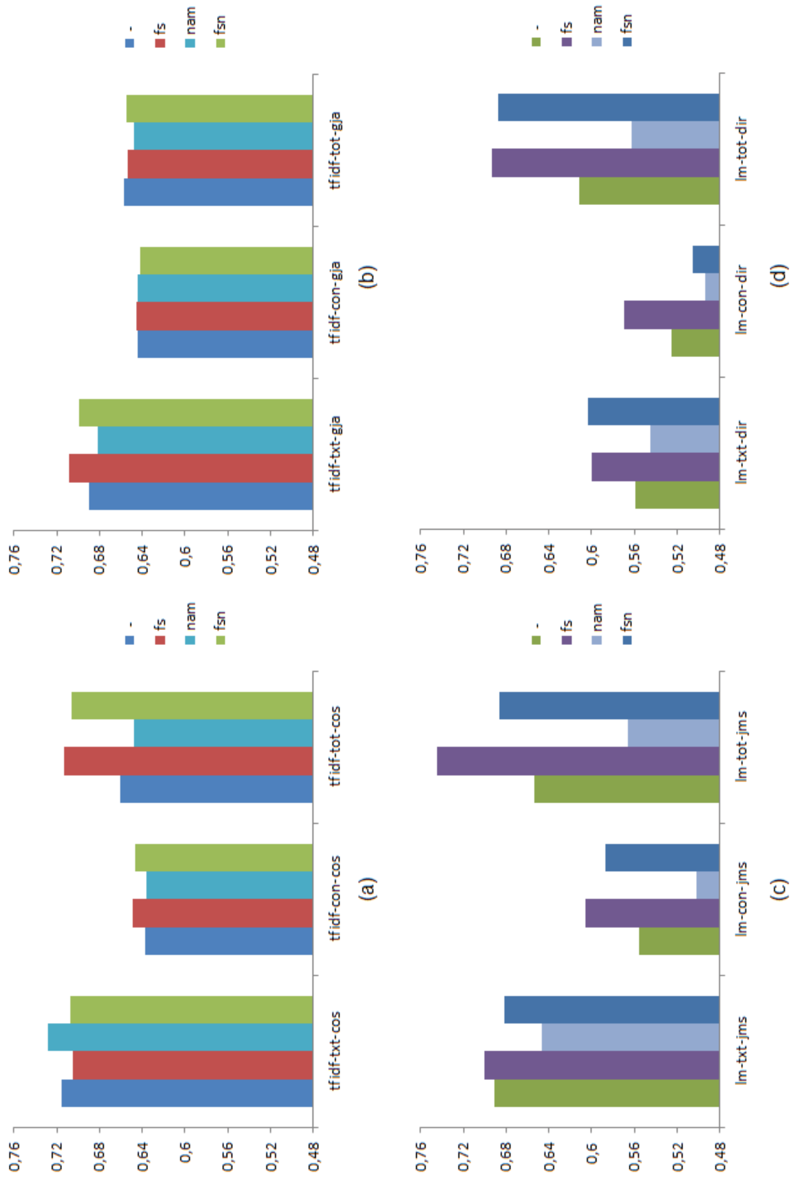


Figure 5.4: Comparison of variations (no variation, *fs*, *nam* or *fsn*) for experiment 2 with a) tf-idf combined with cosine similarity; b) tf-idf combined with generalized Jaccard similarity; c) language modeling with Jelinek-Mercer smoothing; d) language modeling with Dirichlet smoothing. The Y-axis shows the nDCG, while the X-axis indicates the method used.

When both methods based on the vector space model are compared to each other, we see that cosine similarity seems to perform slightly better than the generalized Jaccard similarity. This is interesting since the results of Chapter 4 suggested that the generalized Jaccard similarity outperforms the cosine similarity for research paper similarity, although it also justifies the popularity of this measure. The differences, however, are only significant between *tfidf-tot-cos-fs* and *tfidf-tot-gja-fs* in Experiment 1, and between *tfidf-txt-cos-nam* and *tfidf-txt-gja-nam* in Experiment 2, as indicated by line 1 of Tables 5.8 and 5.9. On the other hand, we can compare both methods based on language modeling. As shown in Table 5.8 and Fig. 5.1, in Experiment 1 they perform comparably for *txt* and *tot*, and only for con methods that use Jelinek-Mercer smoothing clearly outperform those that use Dirichlet smoothing. However, as line 6 of Table 5.8 indicates, there are no significant differences. In Experiment 2 Jelinek-Mercer smoothing seems to perform better also for *txt* and *tot*, meaning a significant improvement in some cases as shown in line 6 of Table 5.9.

Table 5.8: Summary of significant differences based on the method used: *cos*, *gja*, *jms* and *dir*, compared as indicated by the second column; significant difference is indicated by +/- when the first method in the pair performs better/worse than the second one. Experiment 1

		txt				con				tot			
line	method	-	fs	nam	fsn	-	fs	nam	fsn	-	fs	nam	fsn
1	*cos - gja*										+		
2	*cos - jms*			+	+	+	+	+	+	+		+	+
3	*cos - dir*					+	+	+	+				
4	*gja - jms*					+	+	+	+	+		+	
5	*gja - dir*					+	+	+	+				
6	*jms - dir*												

Finally, we compare the methods based on the vector space model with those based on language modeling. In Figures 5.5 and 5.6 we can observe that the former generally outperform the latter. Some methods based on language modeling (*lm-tot-dir-fs*, *lm-tot-dir-fsn* and *lm-tot-jms-fs* in Experiment 1; *lm-tot-jms-fs*, *lm-txt-jms-fs* and *lm-tot-dir-fs* in Experiment 2) perform comparably to those based on the vector space model, but although both vector space model and language model based approaches can achieve good results, the former appear to be much more robust against changes in the particular way in which CFPs are modelled. In a comparison where the information type and the use of feature selection/names is fixed, methods based on the vector space model significantly outperform those based on language modeling in some cases (see lines 2-5 of Tables 5.8 and 5.9). In Experiment 1 these are all cases where *con* is used, plus also some specific

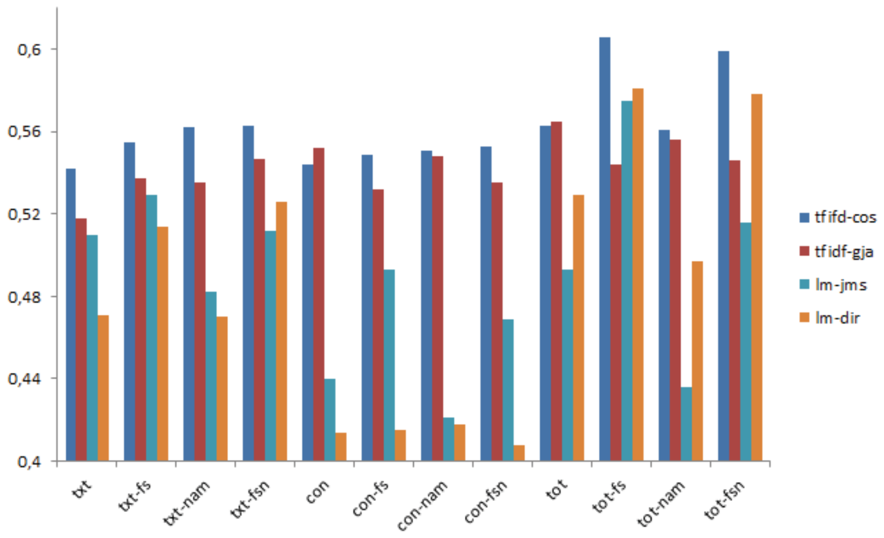


Figure 5.5: Comparison of vector space model based methods and language model based methods for experiment 1. The Y-axis shows the nDCG, while the X-axis indicates the kind of information and variation used.

cases where *txt* or *tot* are used, depending on the compared methods. The differences in the *con* cases are also significant in Experiment 2. This is interesting as the conclusions of Chapter 4 indicated the contrary for the assessment of research paper similarity, but this might be due to the fact that we use the profiles as queries and it has been observed that language models are highly sensitive to smoothing for long and verbose queries [152]. It is also interesting to see that in Experiment 2 methods based on the vector space model always outperform those based on language modeling with Dirichlet smoothing when *txt* is used.

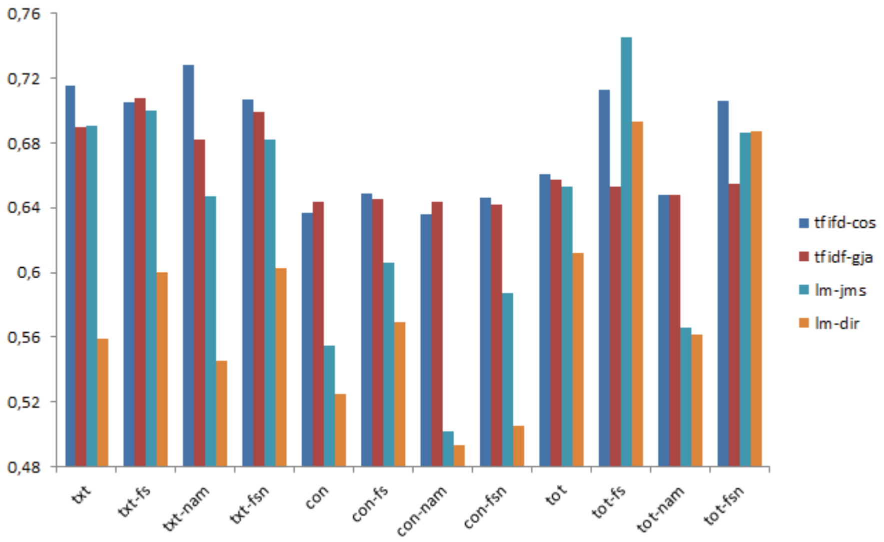


Figure 5.6: Comparison of vector space model based methods and language model based methods for experiment 2. The Y-axis shows the nDCG, while the X-axis indicates the kind of information and variation used.

5.4 Summary

We have proposed and compared several content-based methods to match users with CFPs. We have studied the impact of the different types of information available, the accuracy of the models that represent such information, and the effect of feature selection on these models. Also, using the users' names and the names of the PC members we have accessed the papers recently written by them to profile the users and to complete available information about the CFP respectively. Information about authors frequently cited by the users is also used to reflect the importance given by the users to the CFPs of conferences with people in the PC working in the same field and whose work they usually cite.

The results indicate that methods based on the vector space model are generally more robust, and achieve the best performance on this task. Both for vector space models and language models, feature selection improved the results, which could relate to the problem of having too many terms in the CFPs with a relative low informative load. Since the introductory texts and topics were retrieved automatically, this problem might be alleviated by improving and customizing the parser that performs this task, or by working with clearly structured CFPs, which unfortunately does not occur often in practice.

Finally, we have also seen that although the abstracts of the papers written by the PC members can enhance the performance obtained with the text of the CFP alone they are not powerful enough on their own. This seems to indicate that abstracts contain potentially useful information, but that no method has yet been identified that could fully exploit it. The poor performance of the methods that use abstracts of frequently cited authors might be partly related to this. On the one hand, automatically retrieving the papers written by a particular author is not a trivial problem, as disambiguating author names is a well-studied research area itself. On the other hand, something similar happens with the scope topics: it would be interesting to deal with them separately, but correctly extracting keywords often falls into NLP territory.

As mentioned in the introduction of the chapter, we remark that content-based approaches alone do not suffice to cover all the aspects of CFP recommendation as the relevance of a conference depends also on information not contained in the text of the CFPs. Therefore, the studied content-based methods should be complemented with other techniques. Collaborative filtering would be of great help as it allows using the aforementioned kind of information. In this way, a given CFP can be recommended to a user because another user with similar interests attended a previous edition of that conference. Alternatively, a user can get a recommendation about a given CFP because that conference covers similar topics as a conference he attended in the past. Also, trust-based methods could reflect additional information not covered by collaborative filtering. A user can then be notified about a conference because a researcher he trusts is in the program committee, or because he trusts the conference given its impact on his research field.

Chapter 6

Conclusion

There is an increasing number of online tools on offer to help researchers in their work, from tools aimed at enhancing collaboration to applications to manage specific resources such as project descriptions or scientific papers. In particular, systems dedicated to deal with scientific literature, such as digital libraries and dedicated search engines, have gained much popularity. We are currently witnessing the development of many techniques which take the task of helping the user to find relevant publications a step further: a large number of recommendation methods have been studied in the last years, and now some of them start to get implemented in popular systems.

Within this framework, in this thesis we have presented several methods to filter research resources. On the one hand, we have studied how to exploit various types of information usually found in scientific papers in order to assess the similarity between two papers, and we have proven the suitability of language models for this task. On the other hand, we have explored several content-based methods to recommend calls for papers of conferences (CFPs) based on the different parts of a typical CFP.

In particular, we have first presented a survey of these methods to filter research resources, putting special emphasis on research paper recommender systems since these systems are the most popular ones. This offers a broad perspective of what the state of the art in this research area is right now.

This overview is followed by our main contributions to the domain of filtering of research resources, focusing on content-based approaches. First we have proposed and compared several content-based methods, based either on the vector space model or on language modeling, to compare research paper abstracts. In particular, we have studied how to make best use of semi-structured information about research papers usually accompanying the abstract: a list of keywords, a list of authors, and the name of the journal where the article was published. The results show that the proposed methods perform comparably when only the abstract is considered. However, when the considered semi-structured information (keywords, authors, journal) is employed we can observe that the methods based on language modeling exploit it better and outperform those based on the vector space model. In particular, they interpolate models based on the different types of information available. Also, extra information can be added to the interpolated model by using Latent Dirichlet Allocation (LDA) to discover latent topics and communities. Moreover, the performance of the standard LDA algorithm can be significantly enhanced by using information about keywords and authors associated with a paper for initialization.

On the other hand, we have also proposed and compared several content-based methods to match users with calls for papers of conferences. As with the research papers, we have studied the impact of the different types of information available: the introductory text and list of topics in the scope of the conference, and names of the program committee (PC) members. These names can be used to access the papers recently written by them to complete the information about the CFP. Also, the papers written by the users are retrieved to use them as their profiles. In particular, the abstracts of these papers are assumed to represent their interests, and the citations in these papers can show which authors they usually cite. This latter kind of information can be employed to reflect the importance given by the users to the CFPs of conferences with people in the PC working in the same field and whose work they usually cite. Again, the methods considered are based either on the vector space model or on language modeling. In this context, methods based on the vector space model are generally more robust and achieve the best performance. In both cases feature selection improves the results, and also the performance increases when the abstracts of the papers written by the PC members are considered, instead of taking the text of the CFP alone.

The methods proposed in this thesis are not an endpoint; they offer two main directions for future work. A first possibility is focusing on improving some aspects with a considerable impact on the results and which do not directly relate to the methods but rather to the data used, while the second possibility focuses on exploring alternative methods which can be combined with the proposed ones.

As an example of the first case, we have seen the importance of the information related to the authors, in the case of research paper similarity, or related to the PC members, in the case of the CFP filtering. However, automatically retrieving the papers of a given person is not a trivial task due to the problems mentioned in Chapter 4. Further research to tackle this problem would probably lead to improve the performance of the proposed methods. In general, it is important to ensure the correctness of the data, and a misspelled name is not the only possible error in a document. While research papers usually follow a well-defined structure in which the different parts (keywords, authors, abstract, etc.) are clearly identified, this is not usually the case for CFPs or other documents. In these cases, for example in the case of CFPs, a more robust and complex parser should be used. Also, as mentioned at the end of Chapter 5, some parts of a CFP require something else than a parser, since correctly extracting keywords from the scope topics is in many cases too difficult due to the complex formulation of some sentences. In those cases, techniques from natural language processing should be used. Mitigating these problems would improve the performance of the proposed methods and would also allow to consider new methods based on the proposed ones. For instance, if the keywords could be correctly extracted from the scope topics of the CFPs, they could be used to estimate new models which can later be interpolated as in the case of research paper similarity.

In the second case, and with the goal of effectively applying the proposed methods in a real recommender system, the research should not focus on content-based methods alone, but rather on their integration with other approaches. Collaborative filtering approaches are an ideal candidate, since they have long proven their worth. Also, we have seen that techniques based on citation analysis are popular and can lead to interesting results. Finally, other factors worth to consider in a recommender could be trust (so users who are more trusted by a given user gain more weight in a collaborative filtering approach, for example) or the quality of the papers (e.g. estimated according to the number of papers that cite them, the quality of the journal in which they are published, or more complex authoritativeness indices).

Annexes

Appendix A

An ESA example

In this appendix we show a small example of how the ESA method described in Section 4.2.2 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. A.1, shows the whole process of calculating the ESA vectors for a collection $\mathcal{C}$ of four documents, $\mathcal{C} = \{D_1, D_2, D_3, D_4\}$.

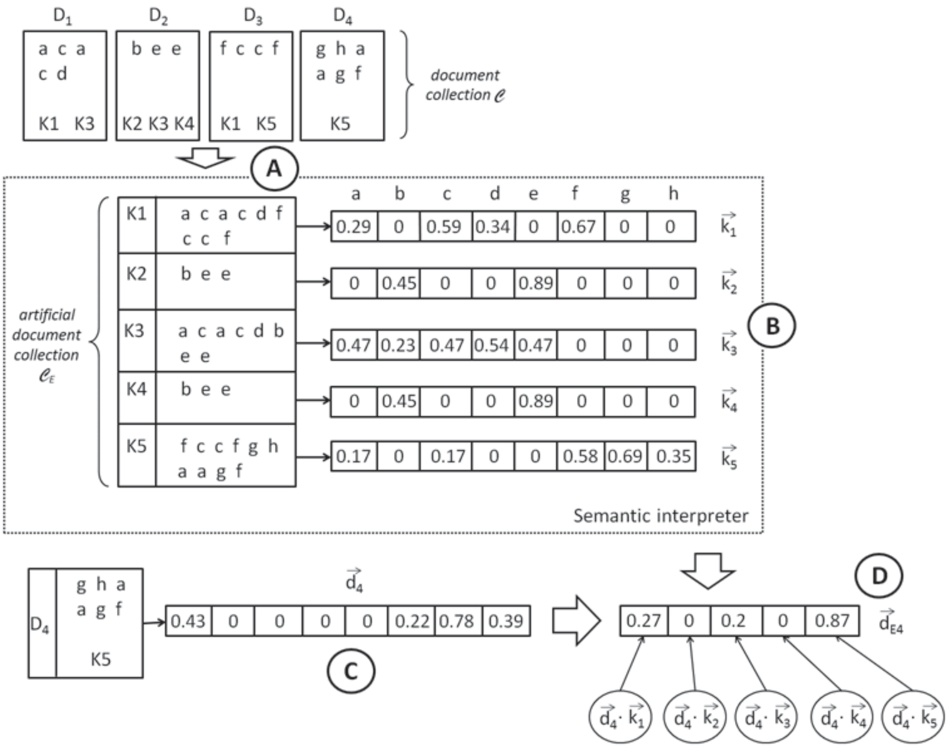


Figure A.1: How ESA vectors are calculated

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 (see Section 2.1.2), 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 of Chapter 4, rather than the quantitative view offered in Section 4.5, 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"*[1].

As explained in Section 4.4, 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.1 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 with the selected paper as for example the first document ranked by *abstract*. On the other hand, the LDA initialization based on keyword clustering causes 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 Eq. (4.18) is big enough).

[1]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.1: 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

Appendix C

Significance values for the experiments with CFPs

In Chapter 5 we summarized the significant differences between the studied methods. In this appendix, we present exact values for the the p obtained with the Mann-Whitney U test for both experiments. Sections C.1 and C.3 refer to Experiment 1, while Sections C.2 and C.4 refer to Experiment 2.

In particular, in Sections C.1 and C.2, for each table, we fix the type of information used, i.e., *txt*, *con* and *tot*, and the variation: none, *fs*, *nam* and *fsn*. On the other hand, in Sections C.3 and C.4, for each table we fix the basic method employed to study the differences between the variations (none, *fs*, *nam* and *fsn*). We highlight in gray those cases in which there is a significant ($p < 0.05$) difference between two methods.

C.1 Experiment 1 - Differences between methods

Table C.1: none-txt

	tfidf-cos	tfidf-gja	lm-jms	lm-dir
tfidf-cos	0	0.3292	0.1587	0.2544
tfidf-gja	0.3292	0	0.3618	0.4209
lm-jms	0.1587	0.3618	0	0.4807
lm-dir	0.2544	0.4209	0.4807	0

Table C.2: none-con

	tfidf-gja	tfidf-cos	lm-dir	lm-jms
tfidf-gja	0	0.596	0.0079	0.0057
tfidf-cos	0.596	0	0.0079	0.0054
lm-dir	0.0079	0.0079	0	0.231
lm-jms	0.0057	0.0054	0.231	0

Table C.3: none-tot

	tfidf-cos	lm-dir	tfidf-gja	lm-jms
tfidf-cos	0	0.4563	0.9415	0.0064
lm-dir	0.4563	0	0.4581	0.3104
tfidf-gja	0.9415	0.4581	0	0.0125
lm-jms	0.0064	0.3104	0.0125	0

Table C.4: fs-txt

	lm-jms	tfidf-cos	lm-dir	tfidf-gja
lm-jms	0	0.3929	0.7092	0.4228
tfidf-cos	0.3929	0	0.3022	0.5144
lm-dir	0.7092	0.3022	0	0.567
tfidf-gja	0.4228	0.5144	0.567	0

Table C.5: fs-con

	lm-dir	tfidf-cos	lm-jms	tfidf-gja
lm-dir	0	0.0059	0.0654	0.0087
tfidf-cos	0.0059	0	0.0447	0.4321
lm-jms	0.0654	0.0447	0	0.0177
tfidf-gja	0.0087	0.4321	0.0177	0

Table C.6: fs-tot

	lm-jms	lm-dir	tfidf-gja	tfidf-cos
lm-jms	0	0.8702	0.2851	0.0727
lm-dir	0.8702	0	0.431	0.6008
tfidf-gja	0.2851	0.431	0	0.0446
tfidf-cos	0.0727	0.6008	0.0446	0

Table C.7: nam-txt

	tfidf-cos	tfidf-gja	lm-dir	lm-jms
tfidf-cos	0	0.3987	0.1296	0.0028
tfidf-gja	0.3987	0	0.3254	0.1782
lm-dir	0.1296	0.3254	0	0.8297
lm-jms	0.0028	0.1782	0.8297	0

Table C.8: nam-con

	lm-dir	lm-jms	tfidf-cos	tfidf-gja
lm-dir	0	0.8332	0.0098	0.0163
lm-jms	0.8332	0	0.0053	0.0083
tfidf-cos	0.0098	0.0053	0	0.8537
tfidf-gja	0.0163	0.0083	0.8537	0

Table C.9: nam-tot

	tfidf-gja	tfidf-cos	lm-jms	lm-dir
tfidf-gja	0	0.776	0.0077	0.2463
tfidf-cos	0.776	0	0.0045	0.2157
lm-jms	0.0077	0.0045	0	0.113
lm-dir	0.2463	0.2157	0.113	0

Table C.10: fsn-txt

	lm-jms	tfidf-gja	lm-dir	tfidf-cos
lm-jms	0	0.115	0.7408	0.038
tfidf-gja	0.115	0	0.5504	0.4703
lm-dir	0.7408	0.5504	0	0.3594
tfidf-cos	0.038	0.4703	0.3594	0

Table C.11: fsn-con

	tfidf-cos	lm-jms	tfidf-gja	lm-dir
tfidf-cos	0	0.0077	0.4248	0.0021
lm-jms	0.0077	0	0.0167	0.0712
tfidf-gja	0.4248	0.0167	0	0.012
lm-dir	0.0021	0.0712	0.012	0

Table C.12: fsn-tot

	lm-dir	tfidf-gja	lm-jms	tfidf-cos
lm-dir	0	0.4937	0.1309	0.624
tfidf-gja	0.4937	0	0.5121	0.0822
lm-jms	0.1309	0.5121	0	0.0259
tfidf-cos	0.624	0.0822	0.0259	0

C.2 Experiment 2 - Differences between methods

Table C.13: none-txt

	tfidf-cos	tfidf-gja	lm-jms	lm-dir
tfidf-cos	0	0.1596	0.2588	0.004
tfidf-gja	0.1596	0	0.9361	0.0129
lm-jms	0.2588	0.9361	0	0.0208
lm-dir	0.004	0.0129	0.0208	0

Table C.14: none-con

	tfidf-gja	tfidf-cos	lm-dir	lm-jms
tfidf-gja	0	0.5504	0.0169	0.0178
tfidf-cos	0.5504	0	0.0214	0.0255
lm-dir	0.0169	0.0214	0	0.1385
lm-jms	0.0178	0.0255	0.1385	0

Table C.15: none-tot

	tfidf-cos	lm-dir	tfidf-gja	lm-jms
tfidf-cos	0	0.2236	0.8583	0.8633
lm-dir	0.2236	0	0.3708	0.1051
tfidf-gja	0.8583	0.3708	0	0.9314
lm-jms	0.8633	0.1051	0.9314	0

Table C.16: fs-txt

	lm-jms	tfidf-cos	lm-dir	tfidf-gja
lm-jms	0	0.7846	0.0028	0.6547
tfidf-cos	0.7846	0	0.0006	0.86
lm-dir	0.0028	0.0006	0	0.0018
tfidf-gja	0.6547	0.86	0.0018	0

Table C.17: fs-con

	lm-dir	tfidf-cos	lm-jms	tfidf-gja
lm-dir	0	0.0749	0.3976	0.1133
tfidf-cos	0.0749	0	0.0783	0.861
lm-jms	0.3976	0.0783	0	0.0072
tfidf-gja	0.1133	0.861	0.0072	0

Table C.18: fs-tot

	lm-jms	lm-dir	tfidf-gja	tfidf-cos
lm-jms	0	0.0129	0.0504	0.1138
lm-dir	0.0129	0	0.3477	0.4061
tfidf-gja	0.0504	0.3477	0	0.082
tfidf-cos	0.1138	0.4061	0.082	0

Table C.19: nam-txt

	tfidf-cos	tfidf-gja	lm-dir	lm-jms
tfidf-cos	0	0.0478	0.0012	0.0067
tfidf-gja	0.0478	0	0.012	0.106
lm-dir	0.0012	0.012	0	0.0554
lm-jms	0.0067	0.106	0.0554	0

Table C.20: nam-con

	lm-dir	lm-jms	tfidf-cos	tfidf-gja
lm-dir	0	0.6199	0.0032	0.0051
lm-jms	0.6199	0	0.0012	0.0026
tfidf-cos	0.0032	0.0012	0	0.6029
tfidf-gja	0.0051	0.0026	0.6029	0

Table C.21: nam-tot

	tfidf-gja	tfidf-cos	lm-jms	lm-dir
tfidf-gja	0	0.9978	0.0438	0.043
tfidf-cos	0.9978	0	0.0113	0.009
lm-jms	0.0438	0.0113	0	0.7645
lm-dir	0.043	0.009	0.7645	0

Table C.22: fsn-txt

	lm-jms	tfidf-gja	lm-dir	tfidf-cos
lm-jms	0	0.4712	0.0079	0.1977
tfidf-gja	0.4712	0	0.0002	0.5646
lm-dir	0.0079	0.0002	0	0.0002
tfidf-cos	0.1977	0.5646	0.0002	0

Table C.23: fsn-con

	tfidf-cos	lm-jms	tfidf-gja	lm-dir
tfidf-cos	0	0.0221	0.8327	0.0045
lm-jms	0.0221	0	0.014	0.0619
tfidf-gja	0.8327	0.014	0	0.0116
lm-dir	0.0045	0.0619	0.0116	0

Table C.24: fsn-tot

	lm-dir	tfidf-gja	lm-jms	tfidf-cos
lm-dir	0	0.4722	0.9477	0.5071
tfidf-gja	0.4722	0	0.5043	0.1486
lm-jms	0.9477	0.5043	0	0.3923
tfidf-cos	0.5071	0.1486	0.3923	0

C.3 Experiment 1 - Differences between variations

Table C.25: tfidf-txt-cos

		-fs	-nam	-fsn
	0	0.6198	0.1956	0.4985
-fs	0.6198	0	0.7724	0.4697
-nam	0.1956	0.7724	0	0.9838
-fsn	0.4985	0.4697	0.9838	0

Table C.26: tfidf-con-cos

		-fs	-nam	-fsn
	0	0.7825	0.5346	0.6127
-fs	0.7825	0	0.9018	0.5453
-nam	0.5346	0.9018	0	0.9056
-fsn	0.6127	0.5453	0.9056	0

Table C.27: tfidf-tot-cos

		-fs	-nam	-fsn
	0	0.1104	0.8409	0.1478
-fs	0.1104	0	0.1145	0.5302
-nam	0.8409	0.1145	0	0.1493
-fsn	0.1478	0.5302	0.1493	0

Table C.28: tfidf-txt-gja

		-fs	-nam	-fsn
	0	0.1375	0.5591	0.3194
-fs	0.1375	0	0.9338	0.6564
-nam	0.5591	0.9338	0	0.2076
-fsn	0.3194	0.6564	0.2076	0

Table C.29: tfidf-con-gja

		-fs	-nam	-fsn
	0	0.4519	0.7065	0.5718
-fs	0.4519	0	0.5002	0.6219
-nam	0.7065	0.5002	0	0.614
-fsn	0.5718	0.6219	0.614	0

Table C.30: tfidf-tot-gja

		-fs	-nam	-fsn
	0	0.2183	0.4056	0.3646
-fs	0.2183	0	0.3508	0.7285
-nam	0.4056	0.3508	0	0.4853
-fsn	0.3646	0.7285	0.4853	0

Table C.31: lm-txt-jms

		-fs	-nam	-fsn
	0	0.082	0.2216	0.8893
-fs	0.082	0	0.111	0.3276
-nam	0.2216	0.111	0	0.1196
-fsn	0.8893	0.3276	0.1196	0

Table C.32: lm-con-jms

		-fs	-nam	-fsn
	0	0.0392	0.2375	0.201
-fs	0.0392	0	0.0546	0.2765
-nam	0.2375	0.0546	0	0.07
-fsn	0.201	0.2765	0.07	0

Table C.33: lm-tot-jms

		-fs	-nam	-fsn
	0	0.0073	0.0335	0.4274
-fs	0.0073	0	0.0112	0.0529
-nam	0.0335	0.0112	0	0.0974
-fsn	0.4274	0.0529	0.0974	0

Table C.34: lm-txt-dir

		-fs	-nam	-fsn
	0	0.351	0.8607	0.3264
-fs	0.351	0	0.3167	0.3585
-nam	0.8607	0.3167	0	0.3
-fsn	0.3264	0.3585	0.3	0

Table C.35: lm-con-dir

		-fs	-nam	-fsn
	0	0.8319	0.8576	0.7103
-fs	0.8319	0	0.9017	0.6678
-nam	0.8576	0.9017	0	0.0647
-fsn	0.7103	0.6678	0.0647	0

Table C.36: lm-tot-dir

		-fs	-nam	-fsn
	0	0.1202	0.2603	0.1036
-fs	0.1202	0	0.1128	0.8017
-nam	0.2603	0.1128	0	0.1029
-fsn	0.1036	0.8017	0.1029	0

C.4 Experiment 2 - Differences between variations

Table C.37: tfidf-txt-cos

		-fs	-nam	-fsn
	0	0.6023	0.1646	0.6696
-fs	0.6023	0	0.1876	0.831
-nam	0.1646	0.1876	0	0.2034
-fs-nam	0.6696	0.831	0.2034	0

Table C.38: tfidf-con-cos

		-fs	-nam	-fsn
	0	0.536	0.9126	0.6727
-fs	0.536	0	0.4311	0.687
-nam	0.9126	0.4311	0	0.588
-fs-nam	0.6727	0.687	0.588	0

Table C.39: tfidf-tot-cos

		-fs	-nam	-fsn
	0	0.0684	0.2383	0.1496
-fs	0.0684	0	0.0394	0.4391
-nam	0.2383	0.0394	0	0.081
-fs-nam	0.1496	0.4391	0.081	0

Table C.40: tfidf-txt-gja

		-fs	-nam	-fsn
	0	0.0988	0.609	0.5853
-fs	0.0988	0	0.0876	0.523
-nam	0.609	0.0876	0	0.2151
-fs-nam	0.5853	0.523	0.2151	0

Table C.41: tfidf-con-gja

		-fs	-nam	-fsn
	0	0.9422	0.9439	0.9177
-fs	0.9422	0	0.9181	0.6931
-nam	0.9439	0.9181	0	0.9312
-fs-nam	0.9177	0.6931	0.9312	0

Table C.42: tfidf-tot-gja

		-fs	-nam	-fsn
	0	0.7804	0.2043	0.9253
-fs	0.7804	0	0.8126	0.7759
-nam	0.2043	0.8126	0	0.7012
-fs-nam	0.9253	0.7759	0.7012	0

Table C.43: lm-txt-jms

		-fs	-nam	-fsn
	0	0.3361	0.0081	0.2561
-fs	0.3361	0	0.0179	0.1595
-nam	0.0081	0.0179	0	0.0314
-fs-nam	0.2561	0.1595	0.0314	0

Table C.44: lm-con-jms

		-fs	-nam	-fsn
	0	0.0639	0.003	0.1716
-fs	0.0639	0	0.0045	0.2168
-nam	0.003	0.0045	0	0.0046
-fs-nam	0.1716	0.2168	0.0046	0

Table C.45: lm-tot-jms

		-fs	-nam	-fsn
	0	0.0006	0.0311	0.0834
-fs	0.0006	0	0.0002	0.0066
-nam	0.0311	0.0002	0	0.0028
-fs-nam	0.0834	0.0066	0.0028	0

Table C.46: lm-txt-dir

		-fs	-nam	-fsn
	0	0.3895	0.2354	0.3428
-fs	0.3895	0	0.2311	0.7786
-nam	0.2354	0.2311	0	0.1932
-fs-nam	0.3428	0.7786	0.1932	0

Table C.47: lm-con-dir

		-fs	-nam	-fsn
	0	0.141	0.1576	0.4258
-fs	0.141	0	0.0158	0.0312
-nam	0.1576	0.0158	0	0.3922
-fs-nam	0.4258	0.0312	0.3922	0

Table C.48: lm-tot-dir

		-fs	-nam	-fsn
	0	0.003	0.1864	0.0007
-fs	0.003	0	0.0038	0.587
-nam	0.1864	0.0038	0	0.0037
-fs-nam	0.0007	0.587	0.0037	0

List of Figures

List of Tables

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. 25

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28, 2009. 26

[3] N. Agarwal, E. Haque, H. Liu, and L. Parsons. Research paper recommender systems: a subspace clustering approach. In *Proceedings of the 6th international conference on Advances in Web-Age Information Management*, pages 475–491, 2005. 32, 34

[4] M. Anderka and B. Stein. The ESA retrieval model revisited. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 670–671, 2009. 21

[5] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *Proceedings of the 17th international conference on World Wide Web*, pages 199–208, 2008. 28

[6] H. Avancini, L. Candela, and U. Straccia. Recommenders in a personalized, collaborative digital library environment. *Journal of Intelligent Information Systems*, 28(3):253–283, 2007. 35

[7] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103, 1998. 19

[8] M. Balabanović. An adaptive web page recommendation service. In *Proceedings of the first international conference on Autonomous agents*, pages 378–385, 1997. 28

[9] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997. 25

[10] K. Balog, L. Azzopardi, and M. de Rijke. A language modeling framework for expert finding. *Information Processing and Management: an International Journal*, 45(1):1–19, 2009. 77

[11] C. Basu, W. W. Cohen, H. Hirsh, and C. G. Nevill-Manning. Technical paper recommendation: A study in combining multiple information sources. *Journal of Artificial Intelligence Research*, 14(1):231–252, 2001. 31

[12] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 714–720, 1998. 28

[13] J. Beel and B. Gipp. Google Scholar's ranking algorithm : an introductory overview. In *Proceedings of the 12th International Conference on Scientometrics and Informetrics*, pages 230–241, 2009. 39

[14] J. Beel and B. Gipp. Google scholar's ranking algorithm: The impact of citation counts (an empirical study). In *Proceedings of the Third International Conference on Research Challenges in Information Science*, pages 439–446, 2009. 39

[15] J. Beel, B. Gipp, S. Langer, and M. Genzmehr. Docear: an academic literature suite for searching, organizing and creating academic literature. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 465–466, 2011. 31, 36

[16] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992. 23

[17] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, 2000. 25

[18] I. Bíró, J. Szabó, and A. A. Benczúr. Latent Dirichlet Allocation in web spam filtering. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, pages 29–32, 2008. 70

[19] A. Birukou, E. Blanzieri, and P. Giorgini. A multi-agent system that facilitates scientific publications search. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 265–272, 2006. 38

[20] D. M. Blei and J. D. McAuliffe. Supervised topic models. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, pages 121–128, 2007. 71

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. 14, 21, 54, 70

[22] T. Bogers. *Recommender Systems for Social Bookmarking*. PhD thesis, University of Tilburg, 2009. 28, 43

[23] T. Bogers and A. van den Bosch. Recommending scientific articles using CiteULike. In *Proceedings of the 2008 ACM Conference on Recommender systems*, pages 287–290, 2008. 31, 34

[24] L. Bolelli, S. Ertekin, D. Zhou, and C. L. Giles. Finding topic trends in digital libraries. In *Proceedings of the Joint International Conference on Digital Libraries*, pages 69–72, 2009. 71

[25] K. D. Bollacker, S. Lawrence, and C. L. Giles. A system for automatic personalized tracking of scientific literature on the web. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 105–113, 1999. 31, 32

[26] K. D. Bollacker, S. Lawrence, and C. L. Giles. Discovering relevant scientific literature on the web. *IEEE Intelligent Systems*, 15(2):42–47, 2000. 32

[27] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52, 1998. 26

[28] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web*, pages 107–117, 1998. 37

[29] R. Burke. Knowledge-based recommender systems. In *Encyclopedia of Library and Information Systems*, 2000. 28

[30] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002. 28

[31] R. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The adaptive web*, pages 377–408. Springer-Verlag, 2007. 28

[32] H.-H. Chen, L. Gou, X. Zhang, and C. L. Giles. Collabseer: a search engine for collaboration discovery. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 231–240, 2011. 38

[33] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, 1996. 13

[34] R. L. Cilibrasi and P. M. Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007. 21

[35] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964. 19

[36] M. Degemmis, P. Lops, and G. Semeraro. A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. *User Modeling and User-Adapted Interaction*, 17(3):217–255, 2007. 25

[37] H. Deng, I. King, and M. R. Lyu. Formal models for expert finding on dblp bibliography data. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 163–172, 2008. 73

[38] H. Deng, I. King, and M. R. Lyu. Enhancing expertise retrieval using community-aware strategies. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1733–1736, 2009. 70

[39] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945. 11

[40] S. T. Dumais and J. Nielsen. Automating the assignment of submitted manuscripts to reviewers. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 233–244, 1992. 31

[41] M. Eirinaki, M. Vazirgiannis, and I. Varlamis. Sewep: using site semantics and a taxonomy to enhance the web personalization process. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2003. 25

[42] G. Erkan and D. R. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, 2004. 19

[43] H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487, 2005. 13

[44] S. Foo and H. Li. Chinese word segmentation and its effect on information retrieval. *Information Processing and Management: an International Journal*, 40(1):161–190, 2004. 19

[45] B. Fuglede and F. Topsøe. Jensen-Shannon divergence and Hilbert space embedding. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 31–36, 2004. 14

[46] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pages 1606–1611, 2007. 9, 21, 47, 50, 51

[47] P. J. Garcés, J. A. Olivas, and 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):564–576, 2006. 72

[48] L. Geng, H. Wang, X. Wang, and 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*, pages 337–346, 2008. 71

[49] B. Gipp and J. Beel. Identifying related documents for research paper recommender by CPA and COA. In *Proceedings of the World Congress on Engineering and Computer Science*, pages 636–639, 2009. 37

[50] B. Gipp, J. Beel, and C. Hentschel. Scienstein: A research paper recommender system. In *Proceedings of the International Conference on Emerging Trends in Computing (ICETiC'09)*, pages 309–315, 2009. 36

[51] S. D. Gollapalli, P. Mitra, and C. L. Giles. Similar researcher search in academic environments. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 167–170, 2012. 38

[52] M. Gori and A. Pucci. Research paper recommender systems: A random-walk based approach. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 778–781, 2006. 31, 37

[53] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982. 19

[54] G. Grefenstette. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Publishers, 1994. 11

[55] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, 2004. 15, 17

[56] U. Hanani, B. Shapira, and P. Shoval. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, 11(3):203–259, 2001. 23, 24

[57] C. S. Hardtke, J. Müller, and T. Berleth. Genetic similarity among arabidopsis thaliana ecotypes estimated by dna sequence comparison. *Plant Molecular Biology*, 32(5):915–922, 1996. 19

[58] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, 1977. 19

[59] A. Hliaoutakis, G. Varelas, E. Voutsakis, E. G. Petrakis, and E. E. Milios. Information retrieval by semantic similarity. *International Journal on Semantic Web and Information Systems*, 2(3):55–73, 2006. 19, 21

[60] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 289–296, 1999. 21

[61] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004. 27

[62] L. Hong and B. D. Davison. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, pages 80–88, 2010. 11, 52

[63] A. Huang. Similarity measures for text document clustering. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, pages 49–56, 2008. 19

[64] W. Huang, S. Kataria, C. Caragea, P. Mitra, C. L. Giles, and L. Rokach. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1910–1914, 2012. 38

[65] G. Hurtado Martín and C. Cornelis. Pas: a personal alert system for information retrieval in criss. In *Proceedings of the Dutch-Belgian Database Day 2007*, 2007. 4

[66] G. Hurtado Martín, C. Cornelis, and H. Naessens. Personalizing information retrieval in criss with fuzzy sets and rough sets. In *Proceedings of the 9th International Conference on Current Research Information Systems*, pages 51–59, 2008. 4, 30

[67] G. Hurtado Martín, C. Cornelis, and H. Naessens. Training a personal alert system for research information recommendation. In *Proceedings of the 13th IFSA World Congress and 6th EUSFLAT Conference*, pages 408–413, 2009. 4, 30

[68] G. Hurtado Martín, S. Schockaert, C. Cornelis, and H. Naessens. Metadata impact on research paper similarity. In *Proceedings of the 14th European conference on Research and Advanced Technology for Digital Libraries*, pages 457–460, 2010. 4

[69] G. Hurtado Martín, S. Schockaert, C. Cornelis, and H. Naessens. Finding similar research papers using language models. In *Proceedings of the 2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation*, pages 106–113, 2011. 4

[70] G. Hurtado Martín, S. Schockaert, C. Cornelis, and H. Naessens. An exploratory study on content-based filtering of call for papers. In *Proceedings of the 6th Information Retrieval Facility Conference*, pages 58–69, 2013. 4

[71] G. Hurtado Martín, S. Schockaert, C. Cornelis, and H. Naessens. Using semi-structured data for assessing research paper similarity. *Information Sciences*, 221(1):245–261, 2013. 4

[72] P. Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901. 11

[73] K. Jack. Mendeley: Recommendation systems for academic literature. `http://www.slideshare.net/KrisJack/mendeley-recommendation-systems-for-academic-literature`. Retrieved May 30, 2013. 45

[74] K. Jack. Recommendation engines for scientific literature. `http://www.slideshare.net/KrisJack/recommendation-engines-for-scientific-literature`. Retrieved May 30, 2013. 45

[75] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989. 19

[76] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, 2000. 81

[77] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980. 70

[78] Y. Jiang, A. Jia, Y. Feng, and D. Zhao. Recommending academic papers via users' reading purposes. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 241–244, 2012. 31, 33, 37

[79] N. Kando, K. Kageura, M. Yoshioka, and K. Oyama. Phrase processing methods for japanese text retrieval. *SIGIR Forum*, 32(2):23–28, 1998. 19

[80] M. Karimzadehgan, R. W. White, and M. Richardson. Enhancing expert finding using organizational hierarchies. In *Proceedings of the 31th European Conference on IR Research*, pages 177–188, 2009. 70

[81] S. Kataria, K. Kumar, R. Rastogi, P. Sen, and S. Sengamedu. Entity disambiguation with hierarchical topic models. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1037–1045, 2011. 71

[82] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25, 1963. 37, 49

[83] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. 33

[84] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer US, 2011. 27

[85] M. Krapivin and M. Marchese. Focused page rank in scientific papers ranking. In *Proceedings of the 11th International Conference on Asian Digital Libraries: Universal and Ubiquitous Access to Information*, pages 144–153, 2008. 37

[86] R. Krestel, P. Fankhauser, and W. Nejdl. Latent Dirichlet allocation for tag recommendation. In *Proceedings of the 2009 ACM Conference on Recommender Systems*, pages 61–68, 2009. 70

[87] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951. 14

[88] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(1):259–284, 1998. 20

[89] M. Lease and E. Charniak. A Dirichlet-smoothed bigram model for retrieving spontaneous speech. In *Proceedings of the Cross-Language Evaluation Forum*, pages 687–694, 2007. 70

[90] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. 19

[91] J. Lewis, S. Ossowski, J. Hicks, M. Errami, and H. R. Garner. Text similarity: an alternative way to search MEDLINE. *Bioinformatics*, 22(18):2298–304, 2006. 19

[92] Y. Liang, Q. Li, and T. Qian. Finding relevant papers based on citation relations. In *Proceedings of the 12th international conference on Web-age information management*, pages 403–414, 2011. 37

[93] H. Lieberman. Letizia: an agent that assists web browsing. In *Proceedings of the 14th international joint conference on Artificial intelligence*, pages 924–929, 1995. 25

[94] T. Liu, S. Liu, and Z. Chen. An evaluation on feature selection for text clustering. In *Proceedings of the 20th International Conference on Machine Learning*, pages 488–495, 2003. 7

[95] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 56

[96] A. S. Lopatenko. Information retrieval in current research information. In *Proceedings of Workshop on Knowledge Markup and Semantic Annotation*, 2001. 30

[97] P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011. 25, 26

[98] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968. 7

[99] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments & Computers*, 28(2):203–208, 1996. 20

[100] P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, 2007. 28, 29

[101] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl. On the recommending of citations for research papers. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, pages 116–125, 2002. 37, 38

[102] S. M. McNee, N. Kapoor, and J. A. Konstan. Don't look stupid: avoiding pitfalls when recommending research papers. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 171–180, 2006. 34

[103] D. M. Mimno and A. McCallum. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 500–509, 2007. 70, 71

[104] B. Mobasher, X. Jin, and Y. Zhou. Semantically enhanced collaborative filtering on the web. In B. Berendt, A. Hotho, D. Mladenič, M. Someren, M. Spiliopoulou, and G. Stumme, editors, *Web Mining: From Web to Semantic Web*, Lecture Notes in Computer Science, pages 57–76. Springer Berlin Heidelberg, 2004. 28

[105] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, 2000. 25

[106] A. Naak, H. Hage, and E. Aïmeur. A multi-criteria collaborative filtering approach for research paper recommendation in Papyres. In G. Babin, P. Kropf, and M. Weiss, editors, *E-Technologies: Innovation in an Open World*, volume 26 of *Lecture Notes in Business Information Processing*, pages 25–39. Springer Berlin Heidelberg, 2009. 35

[107] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550, 2008. 38

[108] M. Ohta, T. Hachiki, and A. Takasu. Related paper recommendation to support online-browsing of research papers. In *Proceedings of the*

Fourth International Conference on Applications of Digital Information and Web Technologies, pages 130–136, 2011. 33, 37

[109] D. O'Sullivan, B. Smyth, and D. C. Wilson. Preserving recommender accuracy and diversity in sparse datasets. *International Journal on Artificial Intelligence Tools*, 13(1):219–235, 2004. 28

[110] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, 2002. 19

[111] D. Parra and P. Brusilovsky. Evaluation of collaborative filtering algorithms for recommending articles on citeulike. In *Workshop on Web 3.0, Hypertext 2009*, 2009. 34, 43

[112] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999. 29

[113] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The adaptive web*, pages 325–341. Springer-Verlag, 2007. 73

[114] M. J. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting web sites. In *Proceedings of the thirteenth national conference on Artificial intelligence*, pages 54–61, 1996. 25

[115] D. Petkova and 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*, pages 599–608, 2006. 13, 70

[116] X. H. Phan, M. L. Nguyen, and 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*, pages 91–100, 2008. 70

[117] J. M. Ponte and 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*, pages 275–281, 1998. 12, 20, 52, 70

[118] X. Quan, G. Liu, Z. Lu, X. Ni, and L. Wenyin. Short text similarity based on probabilistic topics. *Knowledge and Information Systems*, 25:473–491, 2010. 11, 52

[119] I. Rabow. Research information systems in the nordic countries: infrastructure, concepts and organization. research report, Nordbib, 2009. 29

[120] D. Ramage, D. Hall, R. Nallapati, and 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*, pages 248–256, 2009. 71

[121] D. J. Rogers and T. T. Tanimoto. A computer program for classifying plants. *Science*, 132(3434):1115–1118, 1960. 11

[122] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988. 8

[123] G. Salton, E. A. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983. 20

[124] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. 5, 20

[125] M. Sanderson and W. B. Croft. The history of information retrieval research. *Proceedings of the IEEE*, 100(Special Centennial Issue):1444–1451, 2012. 23

[126] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The adaptive web*, pages 291–324. Springer-Verlag, 2007. 26, 27

[127] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proceedings of the 9th Conference on Artificial Intelligence for Applications*, pages 345–352, 1993. 25

[128] N. R. Smalheiser and V. I. Torvik. Author name disambiguation. *Annual Review of Information Science and Technology*, 43(1):1–43, 2009. 72

[129] H. Small. Co-citation in scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973. 37, 49

[130] B. Smyth and P. Cotter. A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000. 28

[131] R. W. Soukoreff and I. S. MacKenzie. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 113–120, 2003. 19

[132] M. Steyvers, P. Smyth, M. Rosen-Zvi, and 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*, pages 306–315, 2004. 71

[133] A. Strehl, E. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search*, pages 58–64, 2000. 19

[134] 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*, pages 705–706, 2007. 38, 73

[135] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4:2–4:2, 2009. 73

[136] K. Sugiyama and M.-Y. Kan. Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 29–38, 2010. 32

[137] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. 15

[138] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl. Enhancing digital libraries with TechLens+. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 228–236, 2004. 33, 34, 35, 37

[139] M. van Setten. *Supporting people in finding information: hybrid recommender systems and goal-based structuring.* PhD thesis, University of Twente, 2005. 28

[140] G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. Petrakis, and E. E. Milios. Semantic similarity methods in wordnet and their application to information retrieval on the web. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 10–16, 2005. 21

[141] A. Vellino. Recommending journal articles with pagerank ratings. Submitted to the Third ACM Conference on Recommender Systems, 2009. 34, 37

[142] R. Vine. Google scholar. *Journal of the Medical Library Association*, 94(1):97–99, 2006. 39

[143] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456, 2011. 36

[144] K. Wegrzyn-Wolska and P. S. Szczepaniak. Classification of rss-formatted documents using full text similarity measures. In *Proceedings of the 5th international conference on Web Engineering*, pages 400–405, 2005. 19

[145] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*, pages 261–270, 2010. 70

[146] W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, 1990. 19

[147] Y. Xu, X. Guo, J. Hao, J. Ma, R. Y. Lau, and W. Xu. Combining social network and semantic concept analysis for personalized academic researcher recommendation. *Decision Support Systems*, 54(1):564–573, 2012. 38

[148] D. Yang and D. M. W. Powers. Measuring semantic similarity in the taxonomy of wordnet. In *Proceedings of the 28th Australasian conference on Computer Science*, pages 315–322, 2005. 21

[149] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, 1997. 7, 8

[150] D. Yarowsky and R. Florian. Taking the load off the conference chairs: Towards a digital paper-routing assistant. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999. 31

[151] K. Yu, J. Lafferty, S. Zhu, and Y. Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1185–1192, 2009. 26

[152] C. Zhai and 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*, pages 334–342, 2001. 13, 70, 93

[153] D. Zhou, J. Bian, S. Zheng, H. Zha, and C. L. Giles. Exploring social annotations for information retrieval. In *Proceedings of the 17th International Conference on World Wide Web*, pages 715–724, 2008. 70

[154] J. Zhu, X. Huang, D. Song, and S. Rüger. Integrating multiple document features in language models for expert finding. *Knowledge and Information Systems*, 23:29–54, 2010. 47, 53