ACADEMIC EXPERTISE REPRESENTATION USING WIKIPEDIA

by

Mahsa Forati

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2016

To my parents.

Table of Contents

List of Tables

List of Figures

Abstract

Finding experts to review a submission or to collaborate with an industry partner is a common problem in the research enterprise that is typically solved manually or by word of mouth. Services like LinkedIn rely on the experts themselves to keep their profiles updated, or the system asks their friends to confirm areas of expertise. The focus of this thesis is on the automatic extraction of expertise representations from the experts' publications, which could be used in a variety of applications such as paper assignment to reviewers in conferences, automatic profile tagging and personalized article recommendation systems.

We are representing expertise areas by a set of computer science research topics defined by Natural Sciences and Engineering Research Council of Canada (NSERC). Each topic is described by a number of keyterms related to different aspects of that topic. We model representing expertise areas of a researcher as a classification problem, where classes are NSERC research topics and instances are researchers. The input of this classifier is a set of features extracted from papers of a researcher and the output is her expertise areas.

To model a researcher, we extract important keyterms from the title and abstract of papers and then find their corresponding concepts and categories in Wikipedia. Keyterm is a word n-gram that explicitly appears in the text. While concepts and categories are the intended meaning of each keyterm without ambiguity. We extract concepts and categories from Wikipedia using different tools like Wikipedia Miner and Sunflower.

We represent documents associated with researchers and research topics in three ways: bag of words, bag of concepts and bag of categories. We calculate the lexical and semantic similarities between a researcher and an NSERC research topic using different methods and use them as input features of the classifier. Then using a labeled dataset first we train our classification model and then test its performance in terms of precision and recall.

Evaluation of this task is not trivial since labeled training data is not readily available. We train and evaluate the system using authors created by gathering conference papers that are on different research topics. We predict the research topic of each author and measure the prediction performance.

List of Abbreviations Used

ACT	Author-Conference-Topic
BOW	Bag of Words
BOC	Bag of Concepts
BOK	Bag of Categories
BPMF	Bayesian Probabilistic Matrix Factorization
COV	Maximum Coverage Algorithm
CBOC	Combined Bag of Concepts
CBOK	Combined Bag of Categories
CQA	Community Question Answering
EFS	Expert Finding System
IBOC	Individual Bag of Concepts
IBOK	Individual Bag of Categories
IDF	Inverse Document Frequency
IP	Integer Programming
LDA	Latent Dirichlet Allocation
LM	Language Model
LR	Linear Regression
LtoR	Learning to Rank
NSERC	Natural Sciences and Engineering Research Council of Canada
POP	Popularity Algorithm
TEM	Topic Expertise Model
TF	Term Frequency
TF-IDF	Term Frequency Inverse Document Frequency
TREC	Text REtrieval Conference

Acknowledgements

I would like to thank my supervisor for everything he has done. Thank you for trusting me and giving me the chance to learn and research under your supervision. Your professional guidance and kind patience made me confident and strengthened me towards facing the problem through this two years. I have learned how to break down big problems and start with simple ideas instead of over-thinking. Thank you for supporting me and providing such a great place to study and research.

I am deeply thankful to my family and friends for their support and help. My deepest thanks to my mom and dad who always supported and encouraged me to pursue my interests. I can never thank you enough for all you have done and your unconditional love. My brothers that there are no limit for their love and kindness. Because I have you, I always have a friend to trust. I am also grateful to my true friends who have always been there for me especially through this year.

Chapter 1

Introduction

There has been a huge demand for finding experts in different areas for years. Traditionally, experts are recognized manually e.g., interviewers assess the expertise areas and the amount of knowledge that somebody has or experts themselves indicate their expertise.

Finding experts is valuable as it leads to better assignment of tasks to the right people and consequently increases the quality of work and decreases the time consumed. In the academic area knowing the research areas of a researcher has different applications as well. First, in conferences we can detect expertise of reviewers, so submitted papers will be assigned to the right matches in a more efficient way. Second, automatically updating profiles of researchers in an institute, could highly improve collaboration among researchers and make better matching between experts and projects. There are many other applications as well such as introducing funding agencies to the appropriate researchers or matching experts for interdisciplinary projects.

With growing community size, detecting expertise areas will become harder and more important. Also, as expertise of people may change over time, updating profiles with the most recent research topics becomes another issue. So we need a system to automatically extract and represent expertise of researchers from documents associated with them.

In this thesis we focus on representing researchers' expertise areas using their publications. Articles published by researchers have an strong connection with their areas of work and knowledge. We propose a classification model to find out the similarity between publications and research topics. Our method captures both lexical and semantic similarities of documents.

Traditional methods such as cosine similarity between TF-IDF vectors are only able to measure explicit overlap between documents based on the shared words. On

the other hand, semantic similarity between two documents could lead to more accurate results by taking into account the meaning of words not just their surface form. However, the semantic similarity is challenging to measure since the topics and concepts are hidden and not mentioned explicitly in the text.

To clarify this importance we show an example of two pieces of text that have the same topic but do not have much lexical overlap. The left text in Fig. 1.1, is about *the effects of a woman's image of her physical shape on eating disorder*. The right text is a paraphrasing[1] of the first i.e., both texts have exactly the same topic. As shown here, two documents could be highly similar without sharing a lot of words together. The opposite is also possible by generating documents that are sharing a lot of words but are in two different topics. However, the latter is less common.

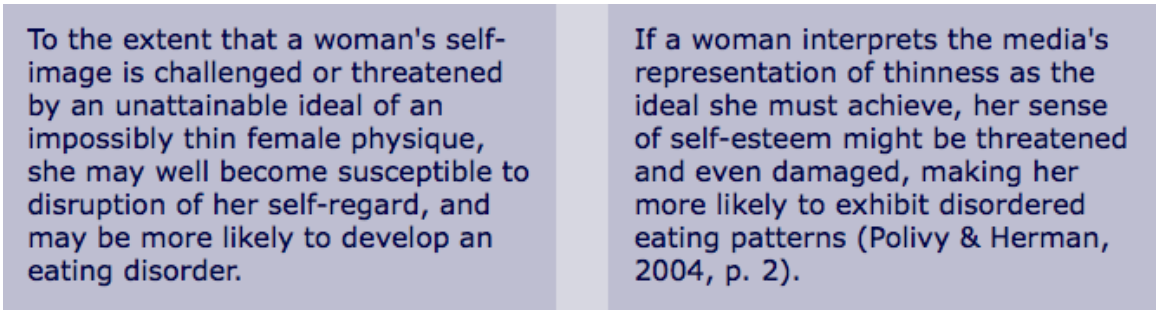

Figure 1.1: A rephrasing example of a text with only a few shared words. These two pieces of text are highly similar yet have low lexical similarity.

Lexical similarity of documents is calculated using bag of words representation. Here, we use the popular TF-IDF vector space model to represent documents related to each researchers. Then the overlap between two documents is calculated using conventional methods e.g., cosine similarity.

The semantic similarity of documents could be measured in different ways. Before measuring the similarity, we need to find hidden concepts in the text. In order to find these concepts, we locate keyterms in the text and map them to their related concepts in Wikipedia.

The meaning of a term is defined within the context and is unique. To represent this unique meaning we use Wikipedia assuming that each article in Wikipedia represents a single concept. For instance, the word "tree" has different articles in

[1] An example of paraphrasing from www.merriam-webster.com/

Wikipedia for each of the meanings as “a plant” and “a data structure”.

With focus of researchers on Wikipedia in the recent years, different similarity measures have been proposed for the relatedness between two concepts. We use the Wikipedia Miner toolkit [Milne and Witten, 2013] to estimate the similarity of concepts. Since each document contains a set of concepts we define different methods to aggregate these scores and calculate the semantic similarity of a researcher’s papers with research topics.

To enrich our document representations we also extract categories using Wikipedia. Categories are extracted for the concepts mentioned in each document. Each category is also a concept but it has a more general meaning. For instance, “plant” is a category for “flower”. These categories provide a different approach to calculate the similarity of researchers with research topics.

We calculate similarity of researchers’ papers with each of the NSERC research topics in many different ways. These similarities form the input of our classifier. NSERC research topics are the classes. By training a classification model, we are able to associate each researcher to the NSERC research topics based on her publications.

Chapter 2

Related Work

In this chapter we review the literature that are related to our work. In the following sections, first we introduce some important work on both expert finding and expertise representation. Since we are using Wikipedia as an external knowledge source to enhance our document representations, we review a few related work in this area, too. Finally, we have a discussion on methods that measure similarity between concepts using Wikipedia.

Before going further we would like to clarify the difference between expert finding and expertise representation which is also discussed in [Balog et al., 2012]. The former is the answer to the question *"Who is expert in topic X?"*, and the latter is the answer to "What are the topics that person Y is expert in?" which is also called the profiling task. The two problems are the same in many aspects, however it is not practical to use the same solution for both of them.

Since the knowledge and expertise are not observable, we have to look for indirect evidence of knowledge that is associated with people. Depending on the context of expert search, this evidence varies. It could be documents associated with people e.g., books, papers, reports and resume, online contents e.g., homepage, blogs and emails or pieces of code written by software developers. Due to the differences in the resources associated with people and availability of these resources, providing a general framework that is applicable on any environment is not easy. Most of proposed solutions are application or domain specific.

2.1 Expert Finding

After announcing expertise retrieval as a new task of Text REtrieval Conference (TREC) in 2005, it received more attention from researchers. To find experts in area X based on their related evidence, two general approaches are commonly used [Balog et al., 2012]. The first approach represents researchers based on their associated

documents and then evaluates the similarity of query with this new representation. This is known as query-independent model. The second approach, also called query-dependent, first locates and ranks documents that are similar to the query. Then, it estimates each expert's score based on their association with the selected documents.

The query independent models tend to be faster and more efficient in terms of data management. Since each expert is summarized and modeled based on the related documents, we are dealing with a significantly smaller model than the original data. However, query dependent models calculate the relatedness of documents on the fly and the impact of every related document is measured, which leads to a more precise query matching [Deng et al., 2008].

Topic modeling has been widely used to extract the existing topics from a collection of data [Blei et al., 2003]. A probabilistic topic modeling method is proposed to extract profiles of researchers from online resources associated with them in the Arnet miner system [Tang et al., 2008]. This model is called Author-Conference-Topic (ACT) and is applicable on papers, authors and publication venues to extract important topics as stated in the paper.

Finding the expertise of people in Community Question Answering (CQA) is also related to our work. Instead of research papers, people's questions and their answers to the other questions are used to extract their expertise. Topic modeling approaches are also useful in finding experts and prompt them to answer questions in CQA [Riahi et al., 2012] and [Yang et al., 2013]. The data gathered from Stack Overflow[1] is used to detect expertise topics and expertise levels of users in both papers. Two topic modeling methods, LDA and STM are used to capture the semantics of a user profile in [Riahi et al., 2012]. LDA combines all posts by a person to extract the topics from a person's profile. STM looks at each post as a separate unit and extracts the topics from each one of them. Moreover, TF-IDF and language model is used to represent the word based profile model of experts.

Besides expert finding, SemEval introduced a task for selecting best answer in CQA in 2015[2]. The methods proposed for this task use lexical features related to the question and answer such as length, words and n-grams. Semantic features also introduced to measure similarity between the question and answer such as features

[1] A popular question and answering website in topics related to computer programming
[2] http://alt.qcri.org/semeval2015/task3/

based on word embedding vectors, language modeling or topic modeling [Alessandro-Moschitti et al., 2015].

One of the systems that performed well in this competition is introduced by [Nicosia et al., 2015]. The semantic features of this system are calculated based on word embedding vectors generated by different approaches. It is also mentioned that word2vec-based [Mikolov et al., 2013] features did not perform well so they were discarded. The next successful system uses word2vec to calculate the similarity between a question and answer. Using doc2vec, they calculate a single vector to represent every question and answer and calculate their cosine similarity. To get better accuracy they also generate the vector representation for every sentence and find the highest cosine similarity between any two sentences of question and answer.

The semantic features that we introduce here are different from these work since we are using Wikipedia as our knowledge source and represent each document as a vector of concepts. We also consider categories that are related to the concepts in the text for calculating the semantic similarity of two documents.

One important application of expert finding in the academic area is matching appropriate reviewers to review submitted papers to a conference or submitted proposals to a grant competition. Currently, this is done manually in most conferences and grant competitions, based on comfort ratings of reviewers, or by manually searching for experts in an expert database. Automatically suggesting suitable pairs of papers and reviewers could find an optimal solution in a shorter time.

Suitability score is defined in [Charlin et al., 2012] as a relevance measure for a given pair of reviewer and paper. A subset of suitability scores are known (manually specified) and the rest of it will be estimated using side information of papers and reviewers. The unseen scores will be learned using three methods of Language Model (LM), Linear Regression (LR) and Bayesian probabilistic matrix factorization (BPMF) from specified scores. In the next stage the problem is formalized as an optimization problem and Integer Programming (IP) is used to solve it. New variations to IP is proposed in this paper to support constraints such as balancing the load on reviewers and assigning sufficient number of reviewers to a paper.

Another approach that fulfills the constraints associated with expert finding is proposed by [Tang et al., 2012]. First, both reviewers' publication papers and submitted

papers are modeled using ACT topic modeling approach. Second, a hybrid similarity score between a given paper and reviewer is defined based on the combination of a language model and the resulting topic model. Finally the matching problem with its constraints is modeled as a minimum convex cost flow which guarantees to find optimal solution.

Textual and semantic concepts associated with a person are not the only resources to use for expertise representation. Other sources such as statistical information based on citation graph and publication records could be used to infer expertise. Besides the textual similarities such as TF, IDF and Okapi BM25 extracted from papers, [Moreira et al., 2015] uses statistical information. Number of publications, average number of publications per year, researcher's h-index and g-index are examples of statistical features. To perform expert finding, a ranking function is learned using *Supervised Learning to Rank (LtoR)* from training data. This function is able to sort experts based on their expertise level regarding the input query. Performance of various *LtoR* approaches are explored in this framework.

CiteseerX[3] is a digital library that automatically extracts and stores articles related to computer and information science. The metadata provided by this library is used to build an expert recommendation system by [Chen et al., 2013]. They use keyphrases extracted from a researcher's papers to retrieve her expertise. A set of candidate keyphrases is built using n-grams of words which appear at least three times in the titles of papers. This set is expanded using Wikipedia hyperlinks. The relevance of a query and a researcher is then estimated by considering lexical relevance (TF) of them. Quality of this estimation is measured based on the citation count of her papers.

There are two important reasons that we did not use Citeseer data. First, this library only contains computer science papers. Second, it does not include information about expertise of authors. Although we have selected computer science research topics to develop our method and perform our experiments, all the steps from gathering data to training model is applicable on other research fields.

[3]http://citeseerx.ist.psu.edu/index

2.2 Expertise Representation

Researchers' papers are usually representative of their expertise and are easily accessible comparing to other resources. Hence, it is common to use information available in papers to represent researchers' profile. Different tools and methods are used to extract information from papers. In most cases, first, important terms from text of papers are extracted. TF-IDF is a traditional criterion to determine terms that are informative in the text and usually used as a baseline for term extraction [Bogers et al., 2006]. Term Frequency (TF) is the number of times a term appears in the text which could be an indicator of its importance. Inverse Document Frequency measures the commonness of term with respect to the whole corpus. The vector of words with the corresponding TF-IDF weights is a representation that reduces a text to a vector of scores. The system proposed by [Ribeiro et al., 2015], builds researchers' profiles by applying algorithms such as TF-IDF, POP and COV [Venetis et al., 2011] on title, abstract and keywords of papers. This work is evaluated through a user study on researchers in Brazil.

Expertise of people changes over time as people may change their research areas or gain more experience in a specific topic. Keeping track of these changes is also another important and yet time taking task. In order to track the changes in an expert's profile [Rybak et al., 2014] developed a hierarchical expertise representation over time. The idea is to find the focus nodes in this hierarchical graph in each timestamp. Shifting focus from a node to another indicates that the expertise area has been changed.

People tend to ask questions from experts, solve colleagues' problems and talk about their areas of interest via emails. Mining topics within emails sent in an organization could help unveiling experts in different areas [Campbell et al., 2003]. Unsupervised clustering is used in this paper to analyze the content of messages. In addition, the relation patterns of emails are used to create the graph of communications to locate and rate experts.

Depending on the application or available data, other sources could also be used for expertise representation. For instance, in a software development company source code developed by individuals could be evidence of their knowledge [Alonso et al., 2008].

2.3 Document Representation

One of the most basic representations of documents is bag of words model which converts the text into a vector of words that could be weighted or unweighted. The popular TF-IDF approach captures importance of each unit in the text. However, treating documents as a set of separated units without considering the relation among words and the concepts behind them, could lead to a poor representation of document and consequently less accurate document similarity measures. In order to compare documents effectively, we need to measure the semantic relatedness between them as well as the syntactical similarity.

Finding the semantics of a document is quite challenging. The semantic units of a document are not explicitly mentioned in the text. So we need to find this information using the evidence exists in the text. Different methods have been introduced to perform this task. Following, we will review two important approaches that are more related to our work.

Topic modeling is a statistical approach to discover topics that are hidden in the document and is being extensively used. This approach has been improved during the years from early work of LSI [Deerwester et al., 1990] and pLSI [Hofmann, 1999] to the popular Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. There are lots of variants to the original LDA that extend it and make it suitable for different applications and frameworks [Tang et al., 2008].

Another approach to model the semantics of a document is to infer topics and concepts based on the terms that appear in the text. It is challenging since first, informative terms should be detected from the text and second, each term should be disambiguated to reflect the right concept behind that term. There are different keyterm extraction systems. Some systems such as Wikipedia Miner [Milne and Witten, 2013] and Illinois Wikifier [Ratinov et al., 2011] wikifies documents i.e., they extract and link terms to the corresponding Wikipedia article. Other systems such as Maui [Medelyan, 2009] uses a controlled vocabulary and only extracts keyterms.

2.4 Concept Similarity Measures

Associating terms with Wikipedia articles in order to represent the intended meaning of that term without ambiguity is a common approach called wikification [Mihalcea and Csomai, 2007]. Wikification means finding the important terms from text and map them to the corresponding Wikipedia article. One approach to measure semantic similarity of documents is to measure similarity of its detected concepts. There are different methods proposed to calculate the similarity of two Wikipedia articles. One of the widely used methods is introduced by [Gabrilovich and Markovitch, 2007]. They represent each word as a weighted vector of Wikipedia articles and then the similarity of each pair of terms in this space is reduced to calculating similarity of two vectors.

Wikipedia as a rich source of knowledge with connected articles and categories in different languages has been used in many different ways. The content of articles, category graph, hyperlink graph and other languages links, all have been used for this purpose. Using the hyperlink structure of Wikipedia articles and by considering amount of common in-links[4] and out-links[5], [Witten and Milne, 2008] defined a relatedness measure for two articles.

Another Wikipedia-based word similarity measure is defined by [Yazdani and Popescu-Belis, 2013]. They explored a random walk algorithm on the Wikipedia article graph to compute the distance between words using the visiting profitability. They also used the hyperlink structure and word co-occurrence information to improve the system performance.

[4]The links that exist in an article and connect different sections of it to other articles in Wikipedia
[5]The incoming links from other Wikipedia articles that points to this article.

Chapter 3

Methodology

In this chapter, we will discuss the details of our system which is designed to automatically extract and represent researchers' expertise based on their publications. The input of our system is titles and abstracts of a researcher's papers and the output is one or more research topics that represents her research areas. We capture the semantic similarity between papers of a researcher and research topics as well as their lexical similarities in order to improve the accuracy.

We are modeling the problem of representing research areas of researchers as a classification problem where research topics are classes and each researcher is an instance. We extract several features from the researcher's papers and feed them to a classifier as input. These features are different similarity scores between a researcher's papers and research topics.

We use the research topics defined by Natural Sciences and Engineering Research Council of Canada (NSERC)[1] to represent the research areas of researchers. NSERC has divided natural sciences and engineering into twelve disciplines and each discipline has between eight to twenty-nine different research topics. Each research topic is described with a set of keyterms. In this thesis we have chosen ten research topics of computer science (see section 4.1.1) to develop and evaluate our method. However, our method is applicable to other topics and disciplines, too.

In our system, each researcher and research topic is represented in three formats: bag of words, bag of concepts and bag of categories. For each researcher, we use these representations to calculate two separate sets of features as illustrated in Fig. 3.1. *Combined papers features* are extracted from all papers combined together as they are a single document. To calculate *Individual paper features*, first we extract intermediate features from each of her publications and then aggregate the results to calculate the final features. We use both sets of features together as the input of our classification

[1] http://www.nserc-crsng.gc.ca/

model. Using a set of labeled training data and these features, we train a classifier.

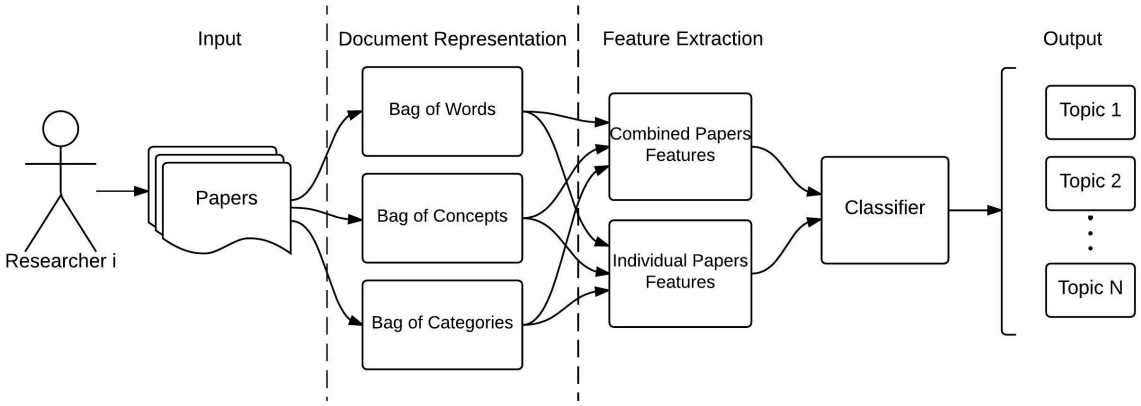

Figure 3.1: From a researcher's papers we extract two sets of features. The first set is calculated based on the combination of all papers into a single document and the second set is calculated using individual papers. Then both sets are combined to represent a researcher and train the classifier.

Note that the total number of features only depends on the number of research topics. Each feature measures the overall similarity between a researcher's papers and a research topic.

In the following sections, first we describe details of different document representations used to represent both researchers and research topics. Next we discuss features extracted from the combined papers of researchers and individual papers.

3.1 Document Representation

To represent a researcher's expertise we need to know how similar her publications are to each research topic. Both researchers and research topics are associated with documents in this system. Each NSERC research topic is described by a set of keyterms. We assume that each topic is a document that consists of its associated keyterms. In order to represent a researcher, we use all of her publications i.e., a set of documents created by concatenation of title and abstract of each paper.

Document similarity is measured in different ways. One common approach is to measure the surface overlap between two documents which is known as lexical similarity. Lexical similarity has been widely used to measure text document similarity. However, it has two disadvantages. The first one is known as vocabulary mismatch.

It happens when the same concept is represented by different terminologies. The second one is ignoring order of words in documents. All words in the text are connected together to form a coherent topic and considering this connection leads to a more accurate similarity measure.

Beyond the lexical similarity of two documents, semantic similarity measures are able to capture the similarity between hidden topics and concepts in the text and uncover the connections between text components. To measure both kinds of similarities we need better ways of representing documents. Hence, we represent researchers and research topics using words that are explicitly mentioned in the text, to capture the lexical similarity, and concepts and categories related to them, to capture the semantic similarity. For this purpose, we introduce bag of words, bag of concepts and bag of categories.

In the rest of this thesis we refer to a word n-gram as a term. The keyterms are the selected terms from the text which consist of up to three words. Each term can have several different meanings and the related meaning could be distinguished considering the context. We call the very unique meaning of a term within the text, concept and we represent each concept using its Wikipedia article.

3.1.1 Bag of Words

Representing documents as bag of words is a traditional approach to measure lexical overlap. As a preprocessing step, we remove stop-words and extract all words from the remaining text. Then each document is represented as a vector of word scores. These scores are a function of Term Frequency (TF) and Inverse Document Frequency (IDF) [Manning et al., 2008]. The pre-processing step and TF calculations are done using Lucene search engine[2] with the standard analyzer.

Each researcher is represented in two ways: individual bag of words and combined bag of words which are used in extraction of individual paper features and combined papers features, respectively. Individual bag of words consists of a set of TF-IDF vectors each one associated with a single paper of the researcher. Combined bag of words is a TF-IDF vector extracted from the combination of individual vectors. Score of each word in this vector equals the average of non-zero values for that word in all

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

individual vectors.

Each NSERC topic is also represented as a binary vector using all words appearing in its keyterms i.e., the score of a word is 1 if it is (part of) a keyterm and it is 0 otherwise. We use the binary vector since we assume that all the keyterms of a topic are important and have equal weight.

3.1.2 Bag of Concepts

Having a good representation of hidden concepts in the text is an initial step towards measuring the semantic similarity. Here we use the Wikipedia articles in order to represent our concepts. Enriching documents with Wikipedia concepts is a popular method which has been used for both classification [Wang and Domeniconi, 2008] [Huang et al., 2012] and clustering [Hu et al., 2008] purposes.

Each article in Wikipedia corresponds to an unambiguous concept and has a unique article ID. Even though Wikipedia covers a large number of concepts, which is about 5 million English articles by the end of 2015[3], not every concept has a corresponding Wikipedia article. Our function is limited to the concepts that appear in Wikipedia.

The task of annotating documents with Wikipedia articles is called wikification [Mihalcea and Csomai, 2007]. Wikification is the process of extracting keyterms from a document and map them to the corresponding Wikipedia articles (Fig. 3.2).

Wikipedia Miner [Milne and Witten, 2013] is a toolkit that offers wikification service. The input is a plain text and output is the annotated text that contains a set of Wikipedia concepts (Fig. 3.3). Moreover, for each detected concept a probability score is calculated that depends on the disambiguation confidence (see Appendix E). We name it *wikification weight* and only accept those concepts that have a *wikification weight* equal to or higher than 0.5. We wikify every paper in the dataset and store the resulting set of concepts. In addition, we extract the associated Wikipedia concepts for each keyterm related to NSERC topics (see Sec. 4.1.1). Our concept space includes the concepts from every paper in the dataset through the wikification process and the concepts that are associated with NSERC topics.

Definition 3.1. *Wikification weight* w_c is a probability score given to every detected

[3]https://stats.wikimedia.org/EN/Sitemap.htm

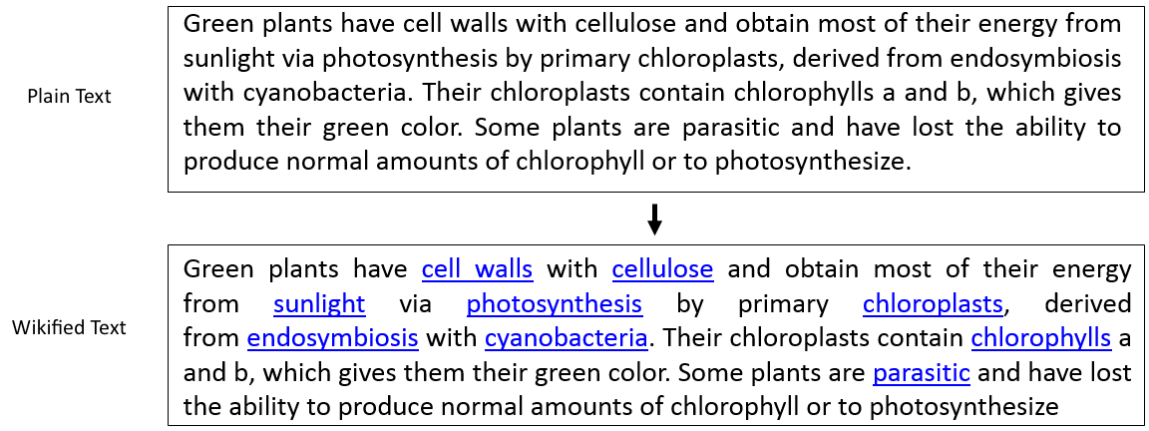

Figure 3.2: The upper box shows a text from Wikipedia in the plain format and the lower one is the wikified result. This example is part of an article from Wikipedia which is manually wikified by its contributors.

Figure 3.3: Input of a wikifier is a plain document and the output is the annotated document containing a set of concepts. Most wikifiers report a probability score for each detected concept. We wikify all papers of a researcher using Wikipedia Miner and stores those concepts that have a score equal to or higher than 0.5.

concept c through wikification process.

Definition 3.2. *Concept space* C is a vector space defined by an alphabetically ordered set of concepts (bag of concepts) derived from union of concepts associated with papers and NSERC research topics. Concepts from papers are extracted through a wikification process. Concepts associated with NSERC research topics are extracted from their keyterms as described in Sec. 4.1.1. Each concept c_i defines a dimension of the concept space.

Each researcher is represented by concepts extracted from her papers in two ways. First, *individual bag of concepts*, which is a set of vectors each one belongs to a single paper of the researcher. Second, a *combined bag of concepts*, a single vector that includes the concepts from all papers of the researcher.

Definition 3.3. *Bag of concepts representation* for a single document d in document set D is a vector $E_d = (w_{c_1,d}, w_{c_2,d}, ..., w_{c_n,d})$ in concept space C with n concepts, where $w_{c_i,d}$ is the weight of concept c_i in document d.

Definition 3.4. *Individual bag of concepts representation* $IBOC_R$ for the researcher R with a set of publications $D_R \subseteq D$ is the set of *bag of concepts representation* of her publications. $IBOC_R = \{E_{d_1}, ..., E_{d_j}, ...E_{d_{|D_R|}}\}$, where $d_j \in D_R$.

Definition 3.5. *Combined bag of concepts representation* of the researcher R, whose publications are all in $D_R \subseteq D$, is a single vector $CBOC_R = (w'_{c_1}, w'_{c_2}, ..., w'_{c_n})$ in concept space C, where w'_{c_i} is the average over the non-zero scores of concept c_i in all publications of researcher R (Eq. 3.1).

$$w'_{c_i} = \underset{w_{c_i,d} \neq 0}{Avg}(w_{c_i,d}), \quad d \in D_R \tag{3.1}$$

For each vector E_d we define two weighting schemes to determine $w_{c_i,d}$ values using binary weights or wikification weights. As a result, each researcher has two representation for each *individual bag of concepts* and *combined bag of concepts* based on these weighting scheme.

Definition 3.6. *Wikification weighting scheme:* In vector E_d that represents bag of concepts of document d, weight of concept c_i equals the wikification weight reported by Wikipedia Miner. $0.5 \leq w_{c_i,d} \leq 1$ if the concept c_i belongs to the concepts of document d and $w_{c_i,d} = 0$ otherwise.

Definition 3.7. *Binary weighting scheme:* in vector E_d that represents bag of concepts of document d, weight of a concept is 1 if it belongs to the concepts of document d and is 0 otherwise.

Definition 3.8. Bag of concepts for each NSERC research topic t is defined as $E_t = (w_{c_1,t}, ..., w_{c_n,t})$, where $w_{c_i,t}$ is a binary weight that equals 1 if concept c_i is associated with topic t and equals 0 otherwise. We only consider binary values for this vector since all of the concepts are assumed to have the same importance.

3.1.3 Bag of Categories

Another representation of documents is bag of categories. Categories of a concept are higher-level concepts in Wikipedia. For instance, the concept "Polar Bear" belongs to the categories "Marine Mammal", "Mammal" and "Animal".

There are several ways to find out the categories of a concept. One trivial solution would be using Wikipedia categories. All Wikipedia concepts have one or more categories that are manually created by different people and they are frequently reviewed by other contributors to correct mistakes and errors. However, they are not necessarily error free and consistent.

Sunflower [Lipczak et al., 2014] is a Wikipedia based generalization system that extracts and represents categories of a concept. Sunflower has three characteristics. First, it uses different languages of Wikipedia to find the most related categories. Second, for each category it calculates the relatedness score. Third, it builds a graph of categories with configurable depth and width. We modified the original Sunflower based on our problem requirements (more details available in Appendix D).

Sunflower's input is a concept and the output is a category graph as shown in Fig. 3.4. This example shows the category graph for the concept "Polar Bear". The number of edges between the input concept and each category in the shortest path shows depth of the category. For each category, a *generalization weight* is reported that measures the relatedness between the input concept and that category.

To represent a document as the bag of categories, all of its concepts extracted in Sec. 3.1.2 are separately given to the Sunflower, one at a time. Bag of categories consists of the union of categories in the resulting category graph of each concept. Each category k_i is extracted by generalizing the concept $c_{i'}$. We refer to $c_{i'}$ as the associated concept with category k_i.

Definition 3.9. [4] *Generalization weight* $G(k_i)$ for category k_i is calculated in the generalization process and shows the similarity between category k_i and its associated concept $c_{i'}$.

Definition 3.10. *Category space* K is a vector space defined by an alphabetically

[4] Some of the definitions in this section is the same as definitions in Sec. 3.1.2. To avoid confusion and to be more clear we repeated them here, too.

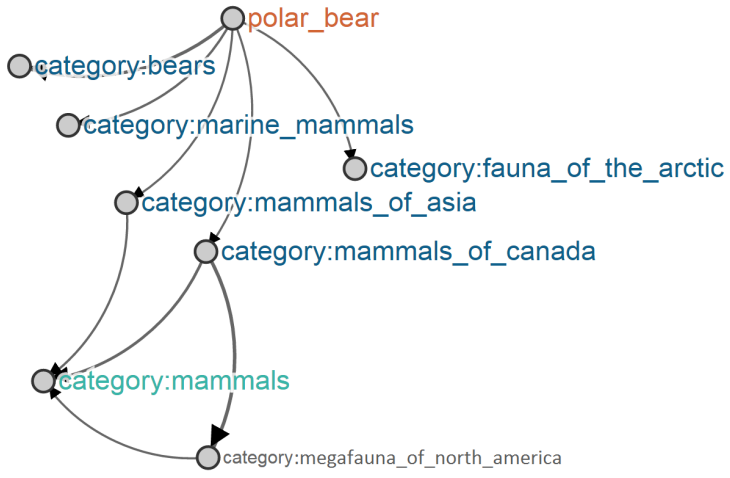

Figure 3.4: The category graph extracted for the concept Polar Bear using Sunflower system.

ordered set of categories (bag of categories) derived from *concept space* C through generalization. Each category k_i defines a dimension of the category space.

We define a researcher's category representations with two formats. First, *individual bag of categories*, which is a set of vectors each one belongs to a single paper. Second, a *combined bag of categories* that is a single vector consists of the categories from all papers of the researcher.

Definition 3.11. *Bag of categories representation* for a single document d in document set D is a vector $V_d = (w_{k_1,d}, w_{k_2,d}, ..., w_{k_m,d})$ in category space K with m categories, where $w_{k_i,d}$ is the weight of category k_i in document d. The categories are extracted by generalizing all concepts belong to document d.

Definition 3.12. *Individual bag of categories representation* $IBOK_R$ of the researcher R with a set of publications $D_R \subseteq D$ is defined as the set of *bag of categories representations* of her publications V_{d_j}. $IBOK_R = \{V_{d_1}, ..., V_{d_j}, ..., V_{d_{|D_R|}}\}$, where $d_j \in D_R$.

Definition 3.13. *Combined bag of categories representation* of the researcher R, with a set of publications $D_R \subseteq D$, is a single vector $CBOK_R = (w'_{k_1}, w'_{k_2}, ..., w'_{k_m})$ in category space K, where w'_{k_i} is the average over the non-zero weights of category k_i in all documents of D_R (Eq.3.2).

$$w'_{k_i} = \underset{wk_{i,d} \neq 0}{Avg}(w_{k_{i,d}}), \quad d \in D_R \tag{3.2}$$

We define three weighting schemes for every vector V_d to determine the values of $w_{k_i,d}$. Generalization, wikification and binary weighting schemes. Using each method, we generate the corresponding category representations of researchers for *individual bag of categories* and *combined bag of categories*.

Definition 3.14. *Generalization weighting scheme:* In vector V_d, weight of category k_i equals the generalization weight calculated by Sunflower $G(k_i)$ multiplied by wikification weight of its associated concept $W(c_{i'})$ (Eq. 3.3).

$$w_{k_i,d} = G(k_i) \times W(c_{i'}), \quad 0 \leq w_{k_i,d} \leq 1 \tag{3.3}$$

Definition 3.15. *Wikification weighting scheme:* In vector V_d, weight of category k_i equals the wikification weight of its associated concept $c_{i'}$ (Eq. 3.4). $0.5 \leq w_{k_i,d} \leq 1$ if the category k_i belongs to the categories of document d and $w_{k_i,d} = 0$ otherwise.

$$w_{k_i,d} = W(c_{i',d}), \quad 0.5 \leq W(c_{i',d}) \leq 1 \tag{3.4}$$

Definition 3.16. *Binary weighting scheme:* in vector V_d weight of a category is equal to 1 if it belongs to the categories of document d and is equal to 0 otherwise.

Definition 3.17. The category vector for each NSERC research topic V_t is extracted by generalizing all of its concepts and is defined as $V_t = (w_{k_1,t}, w_{k_2,t}, ..., w_{k_m,t})$. The w_{k_i} in V_t equals the generalization score $G(w_{k_i})$.

3.2 Feature Extraction

Our feature vector that is calculated for each researcher, consists of different similarity scores with each NSERC research topic. These scores are calculated using different tools and methods and are meant to capture different aspects of similarities. By combining these features using a classifier we will get a single relatedness score for each pair of researcher and research topic.

In the following sections, first we will introduce features extracted from combined papers and then features extracted from individual papers.

3.2.1 Combined Papers Features

To calculate features in this section we use combined *bag of words, bag of concepts and bag of categories* representations. We are calculating four sets of features, where each feature shows the similarity between a researcher and a research topic (Fig. 3.5).

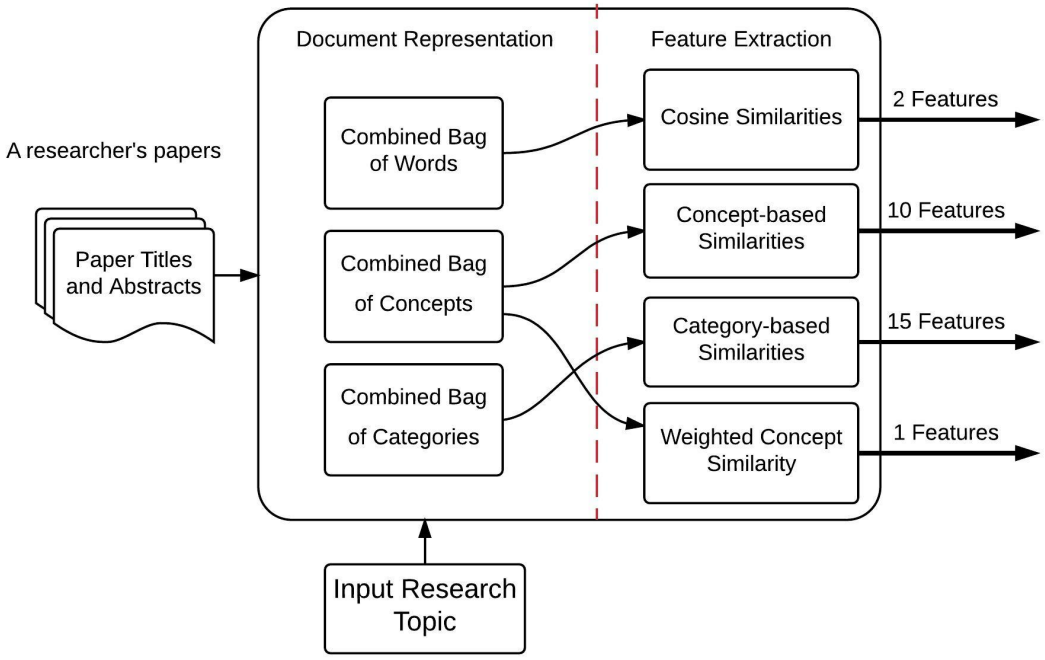

Figure 3.5: Features extracted from combined bag of words, combined bag of concepts and combined bag of categories of a researcher. We calculate four different sets of features based on these representations. Each feature is a similarity score between a researcher and a research topic.

Cosine Similarity

We use TF-IDF vectors to calculate cosine similarity between documents. TF is the number of times a word appears in the text and it is calculated by counting this number for every term. The IDF value is calculated as stated in Eq. 3.5. In this formula the IDF value for word w is calculated while D is the corpus that consists of all documents. The TF-IDF score for each term in the input document is the

multiplication of Term Frequency by Inverse Document Frequency.

$$IDF\,(w, D) = \log \frac{|D|}{|d \in D, w \in d|} + 1 \tag{3.5}$$

The cosine similarity between two vectors of X and Y is calculated using the Eq. 3.6.

$$similarity = \frac{X \cdot Y}{\mid X \mid\mid Y \mid} = \frac{\sum_0^n X_i Y_i}{\sqrt{\sum_0^n X_i^2} \sum_0^n Y_i^2} \tag{3.6}$$

TF is a constant value and it only depends on the document itself. While, IDF could take different values depending on the other documents in the reference corpus. Here we are trying different sets as our reference corpus to capture importance of a word in different settings.

First, we consider all papers in our dataset (see Sec. 4.1.2) which is all of documents in all topics and calculate IDF values. Then we calculate TFIDF vector for every researcher. We define the first feature as:

Definition 3.18. For researcher R and research topic t_i, $F_{C1}(R, t_i)$ is the cosine similarity between combined bag of words of R and bag of words of t_i, where the TF-IDF values of first vector is calculated using the entire corpus D.

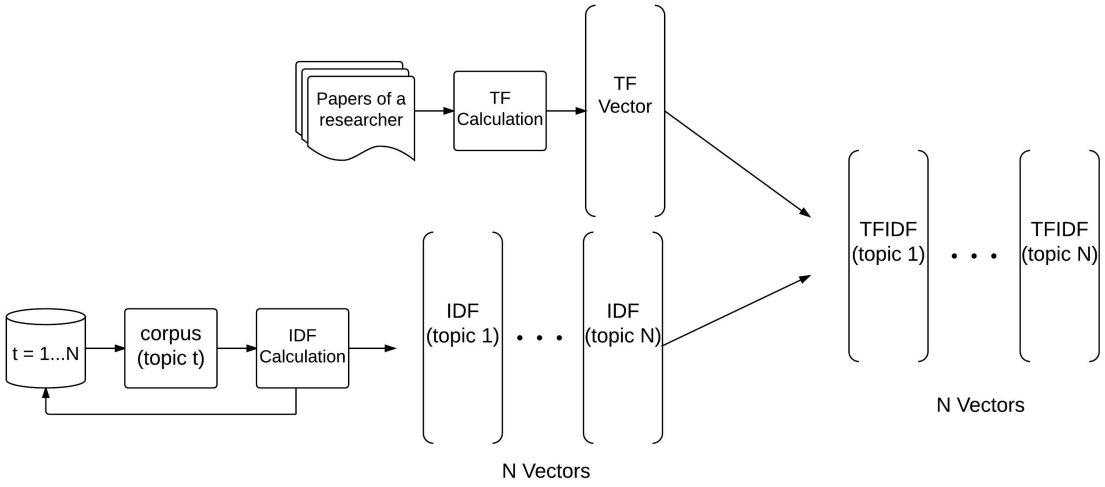

Figure 3.6: We have one TF vector for each researcher and N IDF values for every term in the corpus. So we calculate N different TF-IDF vectors for each researcher.

Second, we take all documents in only one topic as our corpus and calculate IDF values for all words. We repeat this step for each one of our N topics. Finally, we have N different IDF values for each word. The TF vector related to a researcher is constant. So we have N different TF-IDF vectors (Fig. 3.6) representing each researcher and we define the second feature as:

Definition 3.19. For researcher R and research topic t_i, $F_{C2}(R, t_i)$ is the cosine similarity between combined bag of words of R and bag of words of t_i. TF-IDF values of first vector is calculated using the corpus $D_{t_i} \subset D$ where the topic of every paper in D_{t_i} is t_i.

As described, we calculate two cosine similarities for each pair of researcher and research topic using two different TF-IDF vectors related to each researcher (Fig. 3.7).

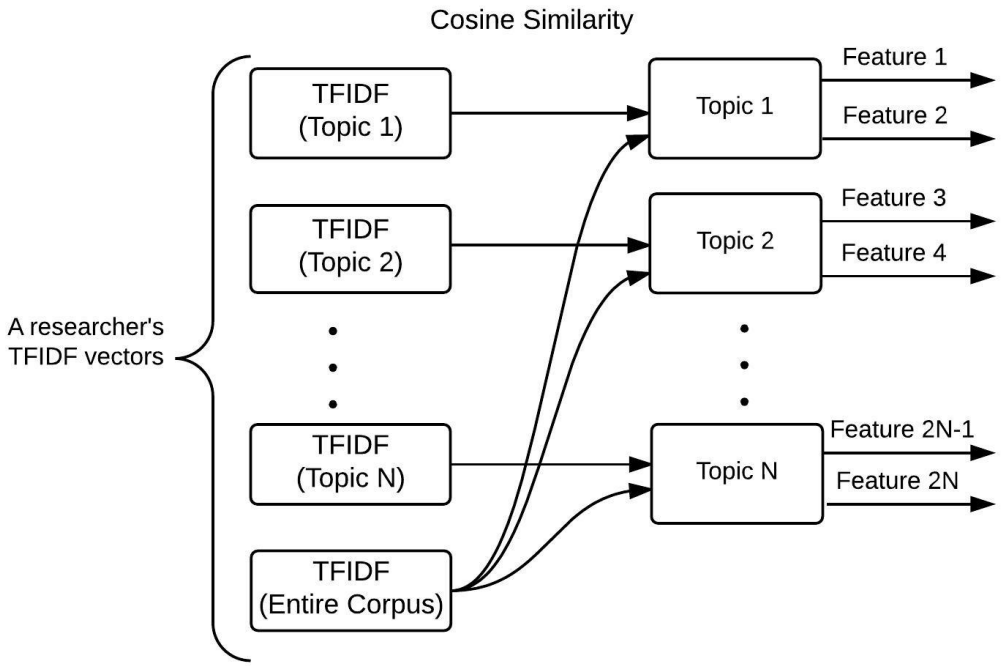

Figure 3.7: Each researcher has $N + 1$ TF-IDF vectors depending on the corpus that is used to calculate the IDF values. One vector is built using the entire dataset and N vectors are associated with each one of the research topics. We calculate two cosine similarities for each pair of researcher and research topic. First using the TF-IDF based on the entire corpus. Second, using the TF-IDF regarding that research topic.

Concept-based Similarity

One approach to measure the semantic similarity of two documents is computing the similarity of their concepts. In this section we propose different methods to measure this similarity. First, we calculate the pairwise similarity of every concepts that we have extracted using Wikipedia Miner toolkit (Eq. 3.7). This service receives two Wikipedia IDs as input and outputs their similarity score. We compute this score for every pair of concepts exist in the corpus.

$$\forall i \in I, j \in J(\forall a \in A_i, \forall b \in B_j \quad \underrightarrow{calculate} \quad Similarity(a,b)) \tag{3.7}$$

In this equation, A is combined bag of concepts for a researcher, B is the bag of concepts for a research topic, I indicates set of researchers and J indicates set of NSERC research topics. Note that in the implementation we do the calculations on the fly and store every new similarity in the look up table. However, we demonstrate the worst case scenario in this section.

We have two weighting schemes for concept vectors as mentioned in Sec. 3.1.2. We calculate the following features for a pair of researcher and research topic based on each weighting scheme. Each feature is calculated for the researcher R and research topic t_i.

1. Cosine similarity of vector of concepts

$$F_{C3} = cosine(CBOW_R, BOW_{t_i}) \tag{3.8}$$

2. Average Wikipedia Miner similarity

$$F_{C4} = \frac{\sum_{a \in A_i} \sum_{b \in B_j} Similarity(a,b)}{|A_i| \times |B_j|} \tag{3.9}$$

3. Maximum Wikipedia Miner similarity

$$F_{C5} = \max_{a \in A_i, b \in B_j} (Similarity(a,b)) \tag{3.10}$$

4. Minimum Wikipedia Miner similarity

$$F_{C6} = \min_{a \in A_i, b \in B_j} (Similarity(a,b)) \tag{3.11}$$

5. Average of maximum Wikipedia Miner similarity

$$F_{C7} = \frac{\sum_{a \in A_i} max\,(Similarity(a,b)) + \sum_{b \in B_j} max\,(Similarity(a,b))}{|A_i| + |B_j|} \quad (3.12)$$

Category-based Features

Our assumption is that two concepts that have common categories are more similar to each other than two concept that does not have any shared categories. So we are using bag of categories representation of documents, to calculate the similarities between them. Each category is a concept, too. So similarity of categories can be calculated the same as the similarity of concepts. Therefore, we calculate the same set of features as we described in Sec. 3.1.2 for categories, too.

First, we calculate the pairwise similarities between categories using Wikipedia Miner (Eq.3.7). Where A is combined bag of categories for a researcher, B is the bag of categories for a research topic, I indicates set of researchers and J indicates set of research topics. Second, we calculate the following features for each researcher[5].

1. Cosine Similarity of vector of categories

$$F_{C8} = \frac{\sum_{a \in A_i} \sum_{b \in B_j} Similarity(a,b)}{|A_i| \times |B_j|} \quad (3.13)$$

2. Average Wikipedia Miner similarity

$$F_{C9} = max\,(Similarity(a,b)) \quad (3.14)$$

3. Maximum Wikipedia Miner similarity

$$F_{C10} = max\,(Similarity(a,b)) \quad (3.15)$$

4. Minimum Wikipedia Miner similarity

$$F_{C11} = min\,(Similarity(a,b)) \quad (3.16)$$

[5] The definitions in this section is the same as definitions in concept-based similarity. To avoid confusion and to be more clear we just repeated them here.

5. Average of maximum Wikipedia Miner similarity

$$F_{C12} = \frac{\sum_{a \in A_i} max_{b \in B_j} (Similarity(a,b)) + \sum_{b \in B_j} max_{a \in A_i} (Similarity(a,b))}{|A_i| + |B_j|} \quad (3.17)$$

Category representation of each researcher have three weighting schemes as described in Sec. 3.1.3. We calculate each feature using all three vectors. Hence, the total number of features in this section equals $5 \times 3 \times N$, where N is the number of research topics.

Weighted Concept Feature

In this section we calculate a new weight for every concept in the bag of concepts representation of each researcher which shows the importance of concept in the document. The idea is to reduce the importance of those concepts that are less relevant to the researchers and have entered to the representation due to keyterm detection, disambiguation or other kinds of errors. We are ranking concepts based on their average relatedness to all other concepts in the text. Similar to the context centrality in [Huang et al., 2012] we are assuming that the concept that is more similar to all other concepts in the text is more important because it could be an indicator of a central topic that other components are there to support these core topics.

The ideal way is to calculate the pairwise similarity of all concepts detected for each researcher as we are representing each researcher as a combined document of her publications. However, because of the inefficiency of this method, we calculate weight of every concept within each individuals paper instead of all papers of a researcher.

As indicated in Fig. 3.8, C1 to C5 are concepts extracted from one paper. We calculate the pairwise similarity between all these concepts using Wikipedia Miner. For each concept, average similarity to other concepts is calculated. For instance for concept C1 we calculate the average value of S1 to S4.

For all concepts in a researcher's papers we calculate this score. If a concept appears in more than one paper, we take the average value as its score. Finally, we have a weighted vector of concepts for each researcher. Using cosine similarity, a feature that shows the similarity of researcher and a research topic is measured. We calculate this feature for all pairs of researcher and research topic.

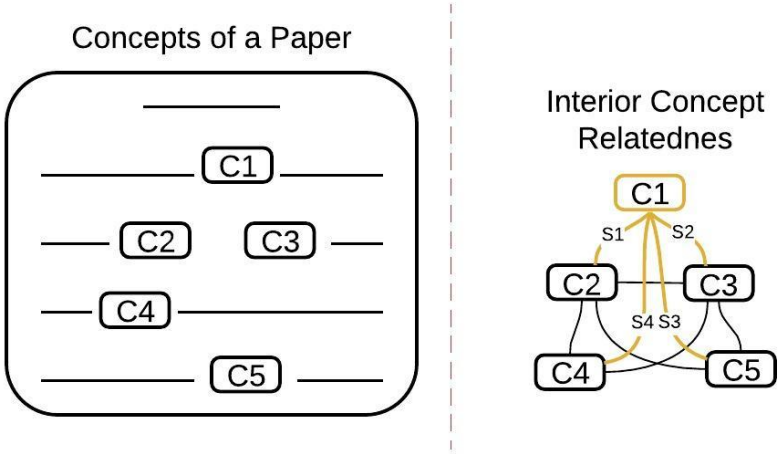

Figure 3.8: Calculating pairwise similarities of concepts in one paper.

3.2.2 Individual Papers Features

In this section, first, we estimate expertise of a researcher based on each of her publications separately. Then we aggregate the results extracted from all papers to calculate the similarity between a researcher and each research topic (Fig. 3.9). For extracting expertise of a researcher based on one paper we use two methods: *1)* cosine similarity *2)* internal classification of individual papers.

Cosine Similarity

In this section we use all three representations of documents i.e., individual bag of words, individual bag of concepts and individual bag of categories. Using each representation, we compute the following features for every pair of researcher and research topic.

First, we calculate the pairwise cosine similarity between each NSERC research topic and each individual paper of researcher R. Second, we aggregate these similarity scores to get a single value for a pair of researcher and research topic. Using these cosine similarities we define two features. The first feature, *aggregated cosine similarity* is the average values of cosine similarities between a research topic and every paper of a researcher. The second feature, *aggregated topic rank* is the average

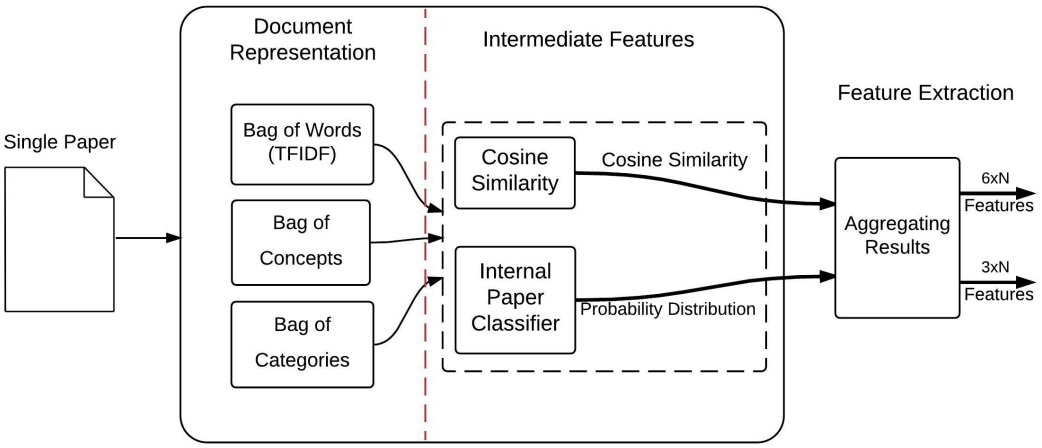

Figure 3.9: Features extracted from Individual papers of a researcher. We use individual bag of words, individual bag of concepts and individual bag of categories to calculate these features. Similarity between a paper and a research topic, is calculated using 1.cosine similarity and 2.internal classifier. We aggregate the results of each method over all papers of a researcher to get a similarity value for a pair of researcher and research topic.

number of times that a research topic has been assigned to a paper. A research topic is assigned to a paper if it has the highest cosine similarity comparing to all other topics.

Definition 3.20. *Aggregated cosine similarity* $F_{I1}(R, t_i)$ between the researcher R and the research topic t_i is the average cosine similarity between all papers d with t_i, where $d \in D_R$ and D_R denotes the set of papers belongs to the researcher R.

Definition 3.21. *Aggregated topic rank* $F_{I2}(R, t_i)$ between the researcher R and the research topic t_i is defined as $F_{I2}(R, t_i) = \frac{|D_{t_i}|}{|D_R|}$, where $D_{t_i} \subseteq D_R$ is the set of papers that are assigned to topic t_i based on the cosine similarity values and T is the set of all NSERC research topics (3.18).

$$\forall d \in D_t \rightarrow \underset{t_i \in T}{argmax}(cosine(d, t_i)) = t \tag{3.18}$$

Calculation of these two features is illustrated in Fig. 3.10. To simplify this example, we assume that the researcher have only 4 papers. Also the total number of research topics equals four. The cosine similarity between a paper and the research topic t_i is denoted by $cosine_{t_i}$ and is shown in the body of that paper.

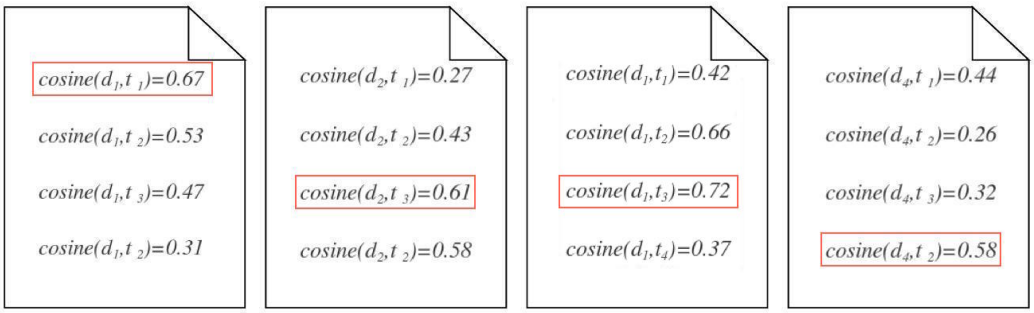

Figure 3.10: In this example cosine similarity between each paper and research topic is shown in the paper. The cosine similarity in the box shows the highest value and the assigned topic to that paper.
$F_{I1}(r_1, t_1) = 0.45, F_{I1}(r_1, t_2) = 0.47, F_{I1}(r_1, t_3) = 0.53, F_{I1}(r_1, t_4) = 0.46$
$F_{I2}(r_1, t_1) = 0.25, F_{I2}(r_1, t_2) = 0.00, F_{I2}(r_1, t_3) = 0.50, F_{I2}(r_1, t_4) = 0.25$

As mentioned earlier, we compute these two features based on all of three document representations. Therefore, for each pair of research topic and researcher we have six features extracted from individual bag of words, individual bag of concepts and individual bag of categories.

Internal Classification of Individual Papers

In the second method we are building an internal classifier which is able to classify each individual paper and assign it to one of the NSERC research topics. Note that this classifier is different from the classifier that we are going to build to classify researchers. In fact, this internal classifier is built and used for the purpose of generating features as the input of outer classifier. The classes of internal classifier are NSERC research topics and instances are individual papers of researchers. The set of features that we use to train the classifier are the same as what we have discussed in Sec. 3.2.1. The only difference is that for this classifier, we use individual bag of words, individual bag of concepts and individual bag of categories to extract features from every paper of the researcher, separately.

In order to build and test the classifier we have randomly selected a subset of research papers in each topic from the original data. We do not use this data later for training or testing in any other part of the system. For classifying papers we have tested three different classification method: random forest, SVM and decision

Classification Method	Precision	Recall	F1-score
Random Forest	**0.555**	0**.556**	**0.526**
SVM	0.524	0.394	0.405
Decision Tree	0.437	0.435	0.436

Table 3.1: Classification performance of single papers using random forest, SVM and decision tree classification methods. We do not use the data used to train this classifier in any other part of the system. Performance measures are reported based on the 10 fold cross validation testing as the results indicate, random forest outperforms the other two models.

tree. The performance of each classifier is tested using 10 fold cross validation and the results are available in Table 3.1. We used Weka[6] implementation to train and evaluate each classifier. Based on the evaluation results we selected random forest as our internal classification method.

The output of random forest for each input instance is a probability distribution over all classes which indicates the probability of being member of each class. Each probability is represented as a number between zero and one and the summation of all of the probabilities equals one. After getting these distributions for each individual paper, we aggregate the results to extract features. For each pair of researcher R and research topic t_i we calculate three features.

Definition 3.22. For researcher R (with the set of papers D_R) and research topic t_i we define: $F_{I3}(R, t_i) = \underset{d \in D_R}{Avg}(p(d, t_i))$. Where $p(d, t_i)$ denotes the probability of topic t_i in the probability distribution calculated based on the classification of paper d.

Definition 3.23. For researcher R and research topic t_i we define: $F_{I4}(R, t_i) = \frac{|D_{t_i}|}{|D_R|}$. Where $D_{t_i} \subseteq D_R$ and $\forall d \in D_{t_i} \rightarrow \underset{t_i \in T}{argmax}(p(d, t_i)) = t_i$. This feature compute the average number of papers in D_R that are assigned to topic t_i.

Definition 3.24. For researcher R and topic t_i we define: $F_{I5}(R, t_i) = \underset{d \in D_{t_i}}{Avg}(p(d, t_i))$. This feature takes an average over class probability values only if $p(d, t_i)$ is the highest value in the probability distribution related to paper d.

Fig. 3.11 is an example to illustrate calculation of these three features. To simplify the example, we assume that researcher has four papers and we have only four classes

[6]http://www.cs.waikato.ac.nz/ml/weka/

as research topics. The probability distribution related to each paper is denoted by P_i and is written in the paper body. $F_{I3}(R, t_i)$, $F_{I4}(R, t_i)$ and $F_{I5}(R, t_i)$ is calculated for the researcher R and each research topic t_i and the values are as following:

1. $F_{I3}(1, 1) = 0.215, F_{I3}(1, 2) = 0.162, F_{I3}(1, 3) = 0.392, F_{I3}(1, 4) = 0.237$
2. $F_{I4}(1, 1) = 0.25, F_{I4}(1, 2) = 0.00, F_{I4}(1, 3) = 0.75, F_{I4}(1, 4) = 0.00$
3. $F_{I5}(1, 1) = 0.35, F_{I5}(1, 2) = 0.00, F_{I5}(1, 3) = 0.466, F_{I5}(1, 4) = 0.00$

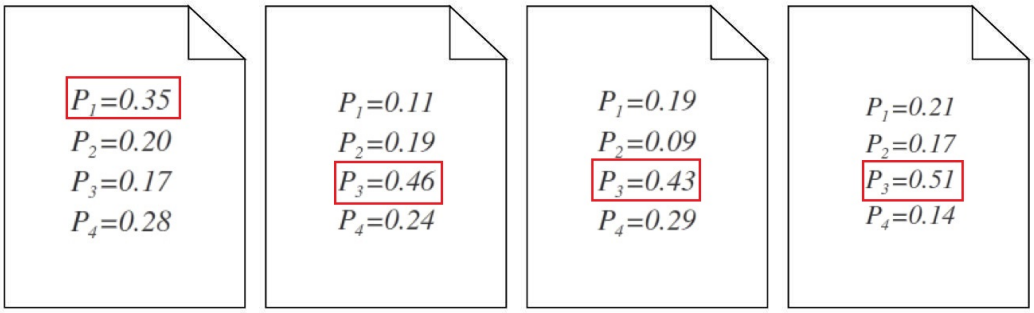

Figure 3.11: This example shows a researcher with four papers and four research topics as classes. Each paper is classified separately and its probability distribution is written in the paper body. P_i is the probability value of research topic t_i.

Chapter 4

Experiments and Results

In this chapter first we describe the data we have used to train and evaluate our model. We discuss the details of gathering and processing data. Then, we compare the results of different classification approaches and the baseline methods and discuss our findings. Finally, to improve the system performance, we conduct a set of experiments including parameter tuning, filtering and feature selection.

4.1 Data Collection

Like most of other data mining and classification problems, finding a good dataset which is labeled and is large enough to be used in a machine learning process is challenging. In order to train our classification model, we need a dataset that includes abstract and title of researchers' papers and their related research areas. As far as we have searched there is no such publicly available dataset, which contains abstract and title of researchers' papers and also research topics of researchers.

4.1.1 Topic Representation

Gathering the required data for training the model needs a lot of time and resources. Thus, we have randomly selected 10 NSERC research topics examples of which are shown in Table 4.1, with each topic described by a set of keyterms (see Appendix A for the full research topics).

Most of these terms are main branches of computer science that have definition pages in Wikipedia. We have access to the titles of Wikipedia pages and developed a simple system which looks through the list of articles and find the ones that are most related to each keyterm by comparing the words available in the keyterm and Wikipedia articles titles. Then we manually select the ones that are most related to each keyterm. We need these concepts to be selected carefully so that they are good representatives of our research topics, however, by skipping the last step we

Topic Title	Keyterms
Web-Enabled Applications and Services	E-health; e-business; e-government; e-learning; e-commerce; e-culture; e-education; e-science; mobile applications
Mathematical Computing	Symbolic computing; scientific computing; numerical optimization; computer algebra; numerical modelling and simulation
Theory of Computing	Theoretical foundations of computation; complexity theory; structural complexity; logic and proof complexity; descriptive complexity; automata theory; information theory; coding theory

Table 4.1: Example list of NSERC research topics used in this thesis. Each topic is described with a set of keyterms.

can automatically match articles which obviously decrease the accuracy. Finally, we have a few Wikipedia concepts for each NSERC research topic that describe different aspects of that topic.

4.1.2 Paper Selection

To create a dataset that consists of authors and their papers, we collect papers from different sources. To gather papers on a specific topic, we use conferences. We choose conferences that are concentrated on only one of the topics and do not have overlap with other topics. In total 43 conferences are selected and from each one, we use a random subset of 20 to 260 papers that were published after 2000. The name of conferences that we use for each topic are shown in Table 4.2. In the first column of this table, we have the topics and in the second column the set of conferences related to each topic. The last column shows the total number of papers selected on that topic.

For collecting these papers, we use the Microsoft Academic API[1] to extract title and abstract of papers and label each paper with the research topic of its conference. At this stage we have a few hundred papers on each topic which we then divide randomly into smaller sets between 30 to 50 papers to create virtual authors. We assume that all papers assigned to a virtual author are on the same topic, however, they are not from the same conference or the same physical author. These virtual authors are

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

Topic	**Conferences**	**Number of Papers**
Mathematical Computing	1. Mathematical Foundations of Computer Science (MFCS) 2. Discrete Mathematics and Theoretical Computer Science (DMTCS)	287
Theory of Computing	1. Annual Symposium on Logic In Computer Science (LICS) 2. Annual ACM symposium on Theory of computing (STOC) 3. Journal of Computer and System Sciences (JCSS) 4. International Symposium on Information Theory (ISIT)	436
Human Computer Interaction	1. International Conference on Advanced Visual Interfaces (AVI) 2. International Conference on Human-Computer Interaction (HCI)	1225

Table 4.2: List of conferences and the total number of research paper gathered for each research topic. Full list of conferences is available in Appendix B

used as the input of our classification model. Four virtual authors, extracted from papers published in three conferences are shown in Fig. 4.1.

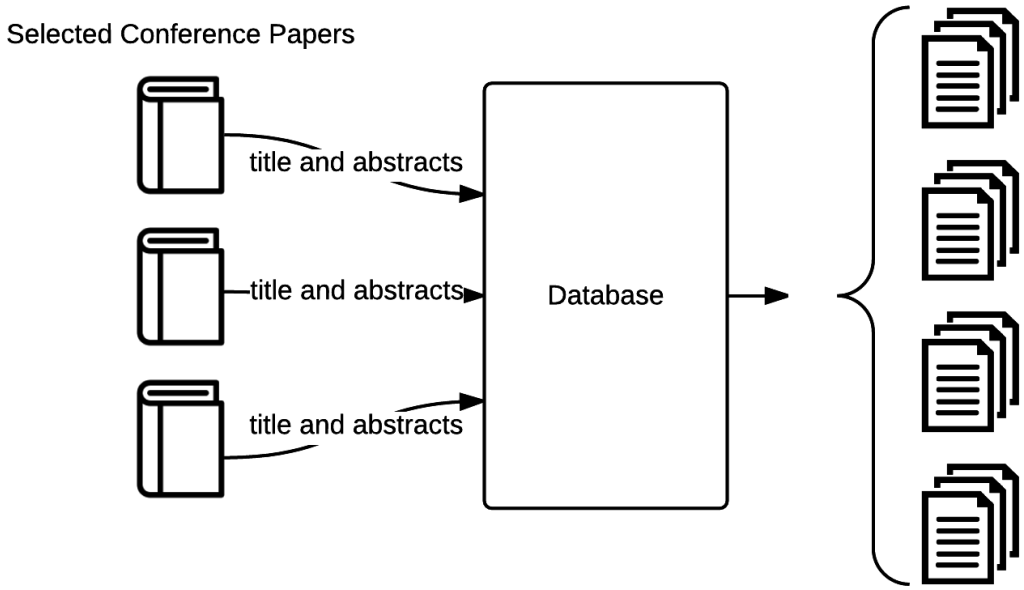

Figure 4.1: Generating virtual authors using papers collected from different conferences. All papers assigned to a virtual author are on the same topic, however, they may not be written by the same physical author.

4.2 Evaluation Measures

In the following experiments we test and report evaluation measures such as precision, recall and F1-score to indicate performance. The precision is calculated based on Eq. 4.1. T_p/ F_P is the number of virtual authors whose expertise topic is correctly/ incorrectly recognized.

$$P = \frac{T_p}{T_p + F_p} \tag{4.1}$$

Recall is defined in Eq. 4.2 where F_n is the number of virtual authors that incorrectly was not recognized with an expertise and T_p is as described previously.

$$R = \frac{T_p}{T_p + F_n} \tag{4.2}$$

The F1-score is the harmonic mean of precision and recall and is calculated as shown in Eq. 4.3. High F1-scores will be achieved only when we have high values for both precision and recall.

$$F_1 = 2\frac{P \times R}{P + R} \tag{4.3}$$

The reported values for precision and recall are all micro-average which means we count the T_p, F_p and F_n for every single class and then calculate the average values for precision and recall. All the values reported for classifiers are the weighted average values based on the output of Weka as we used Weka for all classification tasks.

4.3 Comparing Classifiers and Baselines

In this section, we test different classification methods and compare the effectiveness of our proposed system with the baseline methods. The data we used for training and testing contains 80 virtual authors and 2592 papers in total. In the following experiments 10 fold cross validation is used to evaluate each method.

We test and evaluate four different classification methods on our feature set. All of these classification experiments are done using Weka. The classification models that we train are decision tree, random forest, SVM and multilayer perceptron. Evaluations results of each classifier is available in Table 4.3. For decision tree we used "J48" module of Weka. In the second experiment we used "RandomForest" module where all variables have default values. For SVM classification, we used "LibSVM" method in Weka. We set the cost parameter to 10 and used a linear kernel in this module. Finally, the last experiment is done using "MultilayerPerceptron" function in Weka with default settings for all parameters.

Classifier	Precision	Recall	F1-score
Decision tree	0.715	0.725	0.717
Random forest	0.888	0.875	0.860
SVM	**0.907**	**0.900**	**0.896**
Multilayer perceptron	0.853	0.850	0.847

Table 4.3: Classification performance is measured for different classifiers. The best performance belongs to SVM.

As indicated, the best performance belongs to the SVM classification method.

Because of the nature of our dataset that contains authors who are expert in only one research topic, we train a single label classifier i.e., only one class will be assigned to each virtual author. However, our features are generic and we can train a classifier to detect more than one expertise per author on a training dataset that contains instances with multiple expertise.

The first baseline we test here is based on the cosine similarity between TF-IDF vectors of every virtual author and NSERC research topic pair. We use combined bag of words representation for virtual authors and bag of words representation of NSERC research topics as described in 3.1.1. The similarity between a researcher and a research topic then equals the cosine similarity of their vectors. For each virtual author, the research topic(s) with the highest similarity score is assigned to her. In case that more than one research topics have the same scores or their differences are less than one percent we assign all of them to the virtual author. In the other words, we select the NSERC research topics that have the highest lexical similarity with documents of a researcher as her expertise.

In the second baseline we add the concepts to the vector representation of virtual authors and NSERC research topics. So each virtual author is represented as the union of her combined bag of words and combined bag of concepts using the wikification weighting scheme as discussed in 3.1.2. Each NSERC research topic is also described as the union of its bag of words and bag of concepts vector. Again, both virtual authors and NSERC research topics are described by vectors and we can find a researcher's expertise based on the cosine similarity.

In the third baseline we use Naive Bayes classification model to detect expertise of researchers. The input feature vector is the bag of words of virtual authors. We use Weka to build this input vector. First, we build a document that has a single line per virtual author. This line is the concatenation of titles and abstracts of all of her papers together. Second, we use the "StringToWordVector" option to build the vector representation of researchers. Third we give this representation to Naive Bayes module in Weka and use the 10 fold cross validation to train and test this classifier. The evaluation results of all three baseline methods are available in Table 4.4.

As the evaluation results indicate, our system significantly outperforms all baseline methods. The best classification result in our system belongs to the SVM which

Method	Precision	Recall	F1-score
Cosine Similarity of Word Vectors	0.4895	0.5875	0.5340
Cosine Similarity of Word and Concept Vectors	0.5172	0.5625	0.5389
Naive Bayes Classification	**0.644**	**0.70**	**0.631**

Table 4.4: Evaluation of baseline methods. The first baseline finds the most similar NSERC research topic to each virtual author by calculating cosine similarity of TF-IDF vectors of each pair. In the second baseline we use the union of words and concepts and calculate the cosine similarities. In the last method, we apply a Naive Bayes classifier to the bag of words representation of virtual authors to find the most similar class to them (NSERC research topic).

outperform the best baseline, Naive Bayes with about 0.27 improvement in terms of F1-score. The decision tree classifier has performed as the worst classifier based on our features with the F1-score equals 0.71. However, even the results of this classifier outperforms all of the baseline methods.

4.4 Parameter Tuning

4.4.1 Wikipedia Miner Concept Probability Score

We have used Wikipedia Miner to wikify our documents. This system uses the surrounding words of each keyterm in order to find the right concept for it, however, it is not guaranteed that each keyword match with the right concept. Thus, for each detected keyterm in the text, Wikipedia Miner also reports the probability that the relevant concept is mapped to it (wikification weight).

Concepts with low probability scores are less likely to be accurate. So we ignore all concepts with probability score below a threshold. To find the best value of this threshold, we conduct a set of experiments. These experiments are done on a separate validation dataset which is 20 percent of our original data and contains 20 virtual authors and 730 papers. For each experiment all features with the current configuration have been calculated and the result is based on SVM classification model. We do not use the validation data for training or testing other parts of the system.

We have tried five different values and reported the weighted average precision, recall and F1-score for each value in Table 4.5. As indicated by this experiment the

best threshold equals 0.5.

Threshold	Precision	Recall	F1-score
35	0.633	**0.70**	0.66
40	0.633	**0.70**	0.66
50	0.70	**0.70**	**0.70**
60	**0.73**	**0.70**	0.69
70	0.525	0.60	0.556
80	0.523	0.55	0.519

Table 4.5: The system performance is reported for different values as the threshold of wikification weight.

4.4.2 Sunflower Graph Depth and Width

The Sunflower's output is a graph that shows the related categories of the input concept. This graph is directed and weighted which means each edge has a direction from subcategory to category and each node has a weight that shows its relatedness to the root.

We can configure the width and depth of the output graph in order to avoid getting unrelated or too general categories. The width parameter indicates the maximum number of branches and the depth parameter indicates the maximum level of nodes in the graph. We have tested different values for depth and width to find the best configuration in our problem.

For each of the virtual authors in the validation dataset, we will calculate the entire feature set based on the desired depth and width and run the experiments using SVM classifier. We used 10 fold cross validation for the testing and the weighted average precision, recall and F1-score for the first experiment is reported in Table 4.6.

Width	Depth	Precision	Recall	F1-score
1	1	0.442	0.45	0.43
2	2	**0.633**	**0.75**	**0.677**
3	3	0.623	**0.75**	0.667
4	4	0.583	0.65	0.61
5	5	0.608	0.70	0.643

Table 4.6: Different values of depth and width of Sunflower graph is tested to find the best configuration.

The best configuration so far is depth=2 and width=2. Based on the results of table 4.6 we limit our experiments to the values that either depth or width equal 2. In the next experiment we are testing different values for depth from 3 to 5 when width=2 and different values for width from 3 to 5 when depth=2 and the results are available in Table 4.7.

Width	Depth	Precision	Recall	F1-score
2	5	0.617	0.70	0.65
2	4	0.608	0.70	0.643
2	3	0.592	0.65	0.613
2	2	**0.633**	**0.75**	**0.677**
3	2	0.617	0.70	0.65
4	2	**0.633**	0.70	0.66
5	2	0.625	0.70	0.65

Table 4.7: The system accuracy is reported for different values of depth and width of Sunflower graph when width=2 and when depth=2, respectively.

Based on these experiments and analyzing the trend of F1-score the best choices for Sunflower parameters are width=2 and depth=2.

4.5 Filtering

After visualizing bags of categories for a few virtual authors we noticed that there are unrelated categories among them. We are assuming that if we can find these non relevant categories we could get better results. However, finding a filtering method to detect these words is challenging. Since all the categories that we are using are extracted from Wikipedia, we have tried a few different methods to extract all categories of Wikipedia that are related to computer science. As the results of filtering are not satisfying we put more details about approaches used in the appendix C to simplify this section.

4.6 Feature Selection

Feature selection (also called attribute or variable selection), refers to the process of reducing number of features that are used in the training of a classifier. Since each feature has a different impact on the accuracy of a classifier we can find a good subset of features that are more important and remove the remaining.

Two main benefits of feature selection are: 1. increasing accuracy and 2. decreasing training time. The accuracy of classification could be increased by removing non relevant features or decreasing the importance of less informative features. In addition, as the calculation of some features could be time consuming, lowering the number of features to be calculated will decrease the entire training time, too.

There are different approaches to feature selection. However, finding the optimal subset is a combinatorial problem. Hence, most of the times we are looking for near optimal solutions. We apply three different feature selection methods to our set of features to select the most important ones. These methods are Information Gain, RELIEF and Chi-squared. We use their Weka implementations. The system performance is measured and reported in Table 4.8 using each algorithm. The number of features remaining after applying each method is available in the second column.

Feature Selection Method	Number of Selected Features	Precision	Recall	F1-score
Information Gain	51	**0.916**	**0.913**	**0.912**
RELIEF	116	**0.916**	0.900	0.892
Chi-squared	74	0.904	0.900	0.899

Table 4.8: The performance of system in terms of precision, recall and f-measure after applying three different feature selection algorithms. The number of selected features using each algorithm is available in the second column.

The results of feature selection shows improvement in terms of F1-score. In addition, since the training time for our current dataset is very short (about 2 seconds) we do not measure the improvement of feature selection to this running time. However, we analyzed the top 50 selected features using Information gain, RELIEF and Chi-squared in order to recognize importance of each of the features. We have divided our features into four subsets:

1. Combined word-based features (2 features)

2. Combined concept-based features (10 features)

3. Combined category-based features (15 features)

4. Individual papers features (9 features)

The number of features selected from each group is available in Table reftable:exps-featSel-ddd. Based on the results of this table, the semantic features introduced are highly important comparing to the lexical features.

Feature Set	Information gain	Chi-squared	RELIEF
Combined word-based features	2	4	0
Combined concept-based features	19	19	23
Combined category-based features	15	10	16
Individual papers features	14	17	11

Table 4.9: We analyzed the top 50 selected features using three feature selection method and report the number of features that are selected from each subset.

We conduct a series of experiments to evaluate the importance of each set of features without presence of others. We apply the SVM classification method to each set of features and report precision, recall and f-measure. First, we classify the authors based on combined papers features and individual papers features separately. Next, using each of the bag of words (BOW), bag of concepts (BOC) and bag of categories (BOK) we perform three different experiments. All of these 3 feature sets are a part of combined papers features. The BOW features include all cosine similarities discussed in Sec. 3.2.1. The BOC and BOK features consist of features discussed in concept-based similarity and category-based similarity of Sec. 3.2, respectively. The last feature set we evaluate is the classification of individual papers. Results of all theses experiment are available in Table 4.10.

Feature Set	Precision	Recall	F1-score
Combined papers features	**0.809**	**0.802**	**0.802**
Individual papers features	0.751	0.75	0.747
BOW similarities	0.48	0.542	0.506
BOC similarities	0.749	0.75	0.748
BOC similarities	0.578	0.583	0.576
Individual papers-classification features	0.75	0.74	0.742

Table 4.10: Evaluating performance of the system using different sets of features to measure importance of each set. The first two rows are using combined papers and individual papers features. The next three experiments are done based on the similarity features extracted from each of the bag of words, bag of concepts and bag of categories representation. The last result belongs to the features extracted from classification of individual papers.

Chapter 5

Conclusion

In this thesis we proposed a system that uses researchers papers to characterize their expertise. We used a classification approach to solve this problem. In our classifier, input is the set of titles and abstracts of papers related to a researcher and classes are the research topics. First, we extract different sets of features from papers. Each feature measures the relatedness of a researcher's papers to the research topics. We used different lexical and semantic similarity measures to accurately estimate the research areas of a researcher.

To represent the research topics we used the NSERC categorization model. NSERC divides natural sciences and engineering into twelve disciplines and each discipline is defined by a few research topics. Each research topic is described by keyterms related to different aspects of that topic. To train our classification model and evaluate our system performance we used a subset of research topics in computer science.

We introduce different document representations to calculate similarity between every pair of researcher and research topic. Each document is represented by bag of words, bag of concepts and bag of categories. The bag of words contains all unigrams that explicitly appear in the documents text. The bag of concepts contains the concepts i.e., intended meaning of important keyterms in the document. We use Wikipedia Miner toolkit to extract concepts and wikify documents. We selected Wikipedia to represent concepts because of its rich set of concepts and unique features such as categories and inner connections between articles. The bag of categories is built using Sunflower system. Sunflower is a Wikipedia based system that uses different versions (languages) of Wikipedia to build a category graph for the input concept. We give all concepts extracted from a document to the Sunflower and use the output category graphs to build the bag of categories.

Features are extracted from both individual and combined papers of the researcher. The number of features always depends on the number of research topics.

To calculate combined papers features we take the union of bag of words, bag of concepts and bag of categories for each document that is associated with a researcher. For individual papers features, first we compute similarity of each paper to the classes and then define similarity measures such as average similarity of papers to each class.

Evaluation results show a high accuracy of the system in representing expertise of researchers. We trained and tested different classification approaches using our feature set and reported the results. All of the classifiers trained based on our features significantly outperform the baseline methods.

To improve the system performance we used feature selection algorithms and removed less informative features which increased the accuracy. Analysis of top 50 selected features based on each of the feature selection methods, shows the high impact of semantic features we have introduced. We also conduct a series of experiments to evaluate importance of each set of features without presence of others. Based on the results of feature selection, features extracted from classification of individual papers are at the top of selected feature list and followed by features related to concept similarities extracted from combined papers features. The ranking of features in the results of all three feature selection methods also indicate the importance of semantic similarity between documents comparing to their lexical similarity.

5.1 Future Work

One direction of the future work in this problem is to scale up the system and solve the problem for more than one discipline. Since the research topics that we are using have a hierarchical structure, one possibility is to use hierarchical classification models to scaling up to a dozen disciplines and hundreds of research topics.

Second idea is to use multi-label classifiers with training data where an instance comes with multiple labels. Because of the nature of our dataset we were not able to train a multi-label classifier since all instances have only one class assigned to them. Using a proper dataset that contains instances with more than one class, we are able to use the proposed features to train and evaluate a multi-label model.

In addition to the specific multi-label classification approaches, we also can use one-vs-rest or one-class classification. These methods use a binary approach to decide if an instance belongs to a classes or not and repeat this for all classes. In the

training phase, one-vs-rest takes all instances in a class as positive instances and all other instances as negative. While the one-class classification only use the positive instances that belong to a class and do not use the rest of data as negative instances.

Bibliography

PreslavNakov LlusMarquez WalidMagdy AlessandroMoschitti, James Glass, and Bilal Randeree. Semeval-2015 task 3: Answer selection in community question answering. *SemEval-2015*, page 269, 2015.

Omar Alonso, Premkumar T Devanbu, and Michael Gertz. Expertise identification and visualization from CVS. In *Proceedings of the 2008 international working conference on Mining software repositories*, pages 125–128. ACM, 2008.

Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(2–3):127–256, 2012.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

Toine Bogers, Willem Thoonen, and Antal Van Den Bosch. Expertise classification: Collaborative classification vs. automatic extraction. *Advances in Classification Research Online*, 17(1):1–20, 2006.

Christopher S Campbell, Paul P Maglio, Alex Cozzi, and Byron Dom. Expertise identification using email communications. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 528–531. ACM, 2003.

Laurent Charlin, Richard S Zemel, and Craig Boutilier. A framework for optimizing paper matching. *arXiv preprint arXiv:1202.3706*, 2012.

Hung-Hsuan Chen, Pucktada Treeratpituk, Prasenjit Mitra, and C Lee Giles. CSSeer: an expert recommendation system based on CiteseerX. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 381–382. ACM, 2013.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

Hongbo Deng, Irwin King, and Michael R Lyu. Formal models for expert finding on dblp bibliography data. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 163–172. IEEE, 2008.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

Jian Hu, Lujun Fang, Yang Cao, Hua-Jun Zeng, Hua Li, Qiang Yang, and Zheng Chen. Enhancing text clustering by leveraging Wikipedia semantics. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 179–186, 2008. ISBN 978-1-60558-164-4.

Lan Huang, David Milne, Eibe Frank, and Ian H Witten. Learning a concept-based document similarity measure. *Journal of the American Society for Information Science and Technology*, 63(8):1593–1608, 2012.

Marek Lipczak, A. Koushkestani, and E. Milios. Tulip: Lightweight entity recognition and disambiguation using Wikipedia-Based topic centroids. In *ERD 2014 Challenge, Workshop of the 37th Annual ACM SIGIR 2014 conference*, Gold Coast, Australia, July 6-11 2014.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Olena Medelyan. *Human-competitive automatic topic indexing*. PhD thesis, The University of Waikato, 2009.

Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM, 2007.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

David Milne and Ian H Witten. An open-source toolkit for mining Wikipedia. *Artificial Intelligence*, 194:222–239, 2013.

Catarina Moreira, Pvel Calado, and Bruno Martins. Learning to rank academic experts in the DBLP dataset. *Expert Systems*, 32(4):477–493, 2015. ISSN 1468-0394. doi: 10.1111/exsy.12062. URL `http://dx.doi.org/10.1111/exsy.12062`.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeno, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, et al. Qcri: Answer selection for community question answeringexperiments for arabic and english. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval*, volume 15, pages 203–209, 2015.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.

Fatemeh Riahi, Zainab Zolaktaf, Mahdi Shafiei, and Evangelos Milios. Finding expert users in community question answering. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 791–798. ACM, 2012.

Isac S. Ribeiro, Rodrygo L.T. Santos, Marcos A. Gonçalves, and Alberto H.F. Laender. On tag recommendation for expertise profiling: A case study in the scientific domain. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 189–198, 2015.

Jan Rybak, Krisztian Balog, and Kjetil Nørvåg. Temporal expertise profiling. In *Advances in Information Retrieval*, pages 540–546. Springer, 2014.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, 2008.

Wenbin Tang, Jie Tang, Tao Lei, Chenhao Tan, Bo Gao, and Tian Li. On optimization of expertise matching with various constraints. *Neurocomputing*, 76(1):71–83, 2012.

Petros Venetis, Georgia Koutrika, and Hector Garcia-Molina. On the selection of tags for tag clouds. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 835–844. ACM, 2011.

Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using Wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 713–721, 2008. ISBN 978-1-60558-193-4.

Ian Witten and David Milne. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30, 2008.

Liu Yang, Minghui Qiu, Swapna Gottipati, Feida Zhu, Jing Jiang, Huiping Sun, and Zhong Chen. Cqarank: jointly model topics and expertise in community question answering. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*, pages 99–108. ACM, 2013.

Majid Yazdani and Andrei Popescu-Belis. Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artificial Intelligence*, 194:176–202, 2013.

Torsten Zesch and Iryna Gurevych. Analysis of the Wikipedia category graph for nlp applications. In *Proceedings of the TextGraphs-2 Workshop (NAACL-HLT 2007)*, pages 1–8, 2007.

Appendix A

List of Research Topics and Keyterms

Algorithms and Data Structures: Analysis of algorithms; data structures; parallel and distributed algorithms; graph algorithms; computational combinatorics; computational geometry; randomized algorithms; computational game theory; theoretical cryptography.

Computer Networks: Network protocols; protocol performance; data communications; simulation and emulation of networks; multimedia systems and networks; network management; wireless and mobile networks and ad hoc networks; sensor networks; optical networks; overlay networks and peer to peer networks; information and communication theory; network algorithms; pervasive computing (ubiquitous computing); green networks; cognitive networks; protocol testing.

Information Systems: Models and principles; database systems information; storage and retrieval; information systems; information interfaces and presentations; information integration; visual data analysis; geographic information systems; management information systems; decision support systems; health information systems; medical informatics

Mathematical Computing: Symbolic computing; scientific computing; numerical optimization; computer algebra; numerical modelling and simulation

Theory of Computing: Theoretical foundations of computation; complexity theory; structural complexity; logic and proof complexity; descriptive complexity; automata theory; information theory; coding theory

Software Engineering: Requirements; specification; software design; software

architecture; software implementation; quality management-testing; validation; verification; software development environments; software analysis; evaluation; reliability; maintenance; user interface development; re-engineering and migration; user interfaces; software evolution; process life-cycle models; agile methods; model-driven development; reactive, embedded, and cyber-physical systems; software product lines; data mining from software repositories

Parallel and Distributed Computing: Distributed models and algorithms; distributed architectures; distributed and parallel programming and languages; design, validation and verification; distributed storage; file systems; management; fault tolerance; performance analysis; parallelism and concurrency; parallel processing; parallel models and algorithms; high-performance computing; clusters; symmetric multi-processors; applications; peer-to-peer; grid computing; pervasive computing; map-reduce paradigm; cloud computing; multi-core architectures; service-oriented computing

Web-Based Systems: Social computing; social media; social networks; internet theory; Web services; standards; Web architectural styles (e.g. REST); design of Web systems; Web security; portals and portal frameworks; wikis; blogs; crowdsourcing; recommender systems

Human Computer Interaction: Usability engineering; user interface design and evaluation; multi-modal user interaction; computer-supported cooperative work; haptics; HCI in visualization; virtual reality; human-robot interaction; computer game interfaces; entertainment computing; mixed reality; HCI for mobile devices, modelling/simulation/synthesis of user interfaces

Web-Enabled Applications and Services: E-health; e-business; e-government; e-learning; e-commerce; e-culture; e-education; e-science; mobile applications

Appendix B

List of Conferences

Algorithms and Data Structures: Proceedings of the International Symposium on Algorithms and Computation (ISAAC) - 207

Computer Networks: Computer Communications and Networks (ICCCN), International Conference on Computer Communication (INFOCOM), International conference on networks (ICN), International Conference on Pervasive Computing and Communications (PerCom), International Conference on Network Protocols (ICNP), International Conference on Sensing, Communication and Networking(SECON), Special Interest Group on data Communications (SIGCOMM), International symposium on Mobile ad-hoc networking and computing (MobiHoc) - 546

Information Systems: Proceedings of the International Conference on Information Systems (ICIS), Conference on Research and development in information retrieval (SIGIR), Text REtrieval Conference (TREC), International workshop on Geographic information retrieval (GIR), International conference on Information and knowledge management (CIKM) - 529

Mathematical Computing: Mathematical Foundations of Computer Science (MFCS), Discrete Mathematics and Theoretical Computer Science (DMTCS) - 287

Theory of Computing: Annual Symposium on Logic In Computer Science (LICS), Annual ACM symposium on Theory of computing (STOC), Journal of Computer and System Sciences (JCSS), International Symposium on Information Theory (ISIT) - 436

Software Engineering: International Conference on Software Engineering (ICSE),

Proceedings of the Conference on Software Maintenance (ICSM), International Symposium on Software Testing and Analysis (ISSTA), International Conference on Requirements Engineering (RE), International conference on Automated software engineering (ASE) - 437

Parallel and Distributed Computing: International Conference on Parallel Processing (ICPP), Object Oriented Real-Time Distributed Computing (ISORC), International Conference on Parallel Computing Technologies (PaCT), Parallel Computing: Architectures, Algorithms and Applications (PARCO), International Symposium on Parallel and Distributed Computing (ISPDC), International Conference on Cluster, Cloud and Grid Computing (CCGrid) - 480

Web-Based Systems: Workshop on Web Services and Model-Driven Enterprise Information Systems (WSMDEIS), AAAI CONFERENCE ON WEB AND SOCIAL MEDIA (ICWSM), conference on Hypertext and hypermedia (HT),international conference on World wide web (WWW) - 359

Human Computer Interaction: International Conference on Advanced Visual Interfaces (AVI), International Conference on Human-Computer Interaction (HCI) - 1225

Web-Enabled Applications and Services: International Conference on Advances in Web-Based Learning (ICWL), International Conference on Web Services (ICWS), Commerce and Enterprise Computing (CEC), International Conference on e-Technology, e-Commerce and e-Service (EEE), International conference on Electronic commerce (ICEC), International conference on Electronic Government (EGOV), Conference on Electronic Commerce (EC) - 436

Appendix C

Filtering

The cloud visualization of bag of top categories for a researcher who is expert in computer networks is available in Fig. C.1. There are some irrelevant categories such as "latin words and phrases", "history" and "2001".

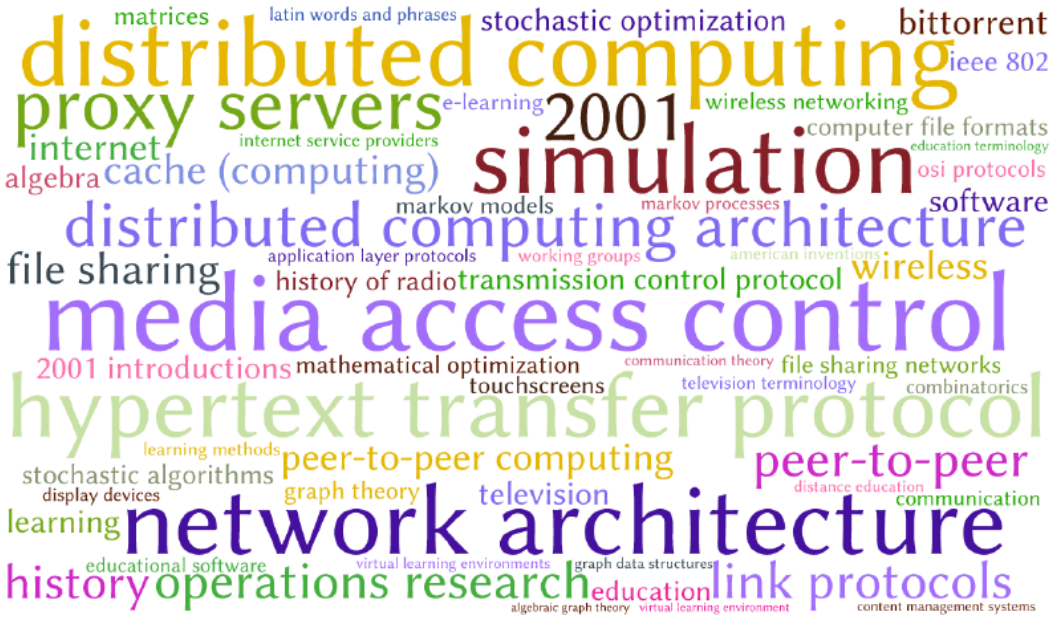

Figure C.1: Word cloud representation of bag of categories for a researcher who is expert in Computer Networks. The size of each word corresponds to its weight in the categories vector of the author.

One trivial method to filter unrelated categories is retaining all categories that are connected to computer science and remove everything else. For this purpose, first we use the Wikipedia category graph. Wikipedia categories have a graph structure in which categories are connected to their subcategories. This graph has both cycles and isolated nodes [Zesch and Gurevych, 2007], too. As we are looking for all categories related to computer science, we find the "Computer Science" category node in this graph and extract all of its subcategories. So we have a connected graph with

computer science as its root.

After analyzing the result graph we found lots of nodes that are not related to computer science. They have been added to the graph because of either human errors while editing Wikipedia or ambiguity of categories. A part of this graph hierarchy is available in Fig. C.2. Because of the ambiguity of "bio-informatics" category, two unrelated branches are merged together. At some point in this graph categories such as "Nervous System", "Senses" and "Organ System" has been added to the filtering list which are not relevant.

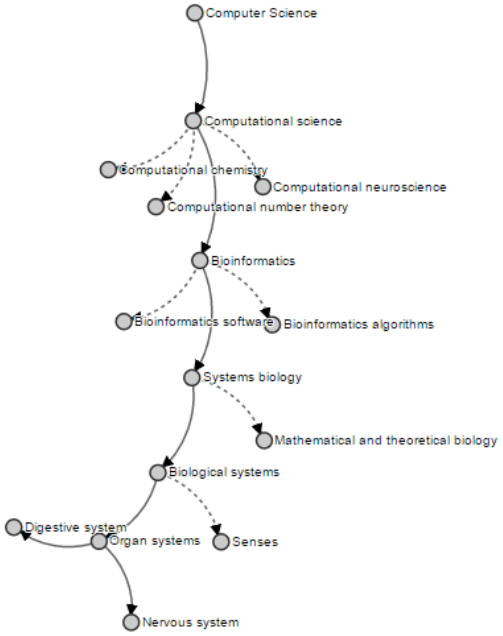

Figure C.2: A sub-graph extracted from the first filtering graph. This graph is constructed by subcategories of computer science, however, a lot of unrelated nodes exist in it. This is mostly because of ambiguity of categories or sometimes because of human mistakes.

To improve the previous graph we try two approaches to find unrelated nodes and remove them. First, we find related categories based on their similarity to the root node. We use Wikipedia miner similarity score and calculate the pairwise similarity of every category in the graph with the computer science. Second, we measure similarity of every category with its direct parent. The average value of these two scores shows its relatedness to the computer science. Then we filter out unrelated categories in two ways: 1. Removing only the node that its score is lower than a specific threshold. 2.

Pruning the whole branch from the node that has a score lower than the threshold. Using each of these methods we will have a category list that are more likely to be related to computer science.

Using all categories in the filtering graph, we filter bag of categories representation of researchers i.e., removing any category that exists in bag of categories but does not appear in the filtering graph. After evaluating the system using each of the filtering methods, the system performance decreased in terms of precision and recall. Analyzing the results of bag of categories, before and after filtering shows that filtering removes a lot of categories that are informative and retains a portion of non related categories, too (see Fig. C.3).

Figure C.3: An bag of categories example for and expert in software engineering topics. The left picture shows top categories before filtering and the right one shows the results after filtering using the first filtering model.

Appendix D

Sunflower

Wikipedia is a large multilingual encyclopedia that has about 38 million pages as of December 2015 and new contents are added each day. It currently has 290 active languages growing around the world. Articles are created, edited and maintained by volunteer contributors. A Wikipedia article doesn't necessarily have a unique structure, however, all articles have an id, title, content, references and one or more categories.

Sunflower is an open sourve system developed in Dalhousie university with the purpose of generalizing concepts and finding their categories. It was initially designed to be used for generalization of named entities. The basic idea of Sunflower is to use Wikipedia in different languages in order to find categories of a concept. It uses Wikipedia since each article in Wikipedia is associated with a single concept.

One trivial approach to this problem is using the provided categories for each article to generalize it. Unfortunately, not all of the Wikipedia categories are informative for the purpose of logical categorization. For example there are a large number of categories such as "Wikipedia articles in need of updating", "Wikipedia Bad Jokes and Other Deleted Nonsense", "Wikipedia backlog" and "Wikipedia featured articles" that does not have information about that concept. Moreover, articles do not necessarily have a heterogeneous categories list in terms of number of related categories that are covered and the generality levels of categories.

The key idea of Sunflower is using different languages of Wikipedia to generalize a concept. To decide if an article belongs to a category or not, Sunflower takes each version (language) of that article as an evidence. In addition, the categories in Wikipedia have a hierarchical structure. The graph of categories in each language has a unique structure. Sunflower finds the graph of categories for an article in each language and then unifies these graphs to build a single output graph. The probability that article a belongs to the category c is calculated as Eq D.1 where L is all articles

in one language of Wikipedia.

$$probability(a, c) = \frac{|\{a \in L, c \in a\}|}{| \ a \in L|} \tag{D.1}$$

To find categories of a concept, Sunflower extracts all category information of articles and builds a lookup table. A part of the table is available in Fig. D.1. In the first column the article title is shown and in the second column the number of languages that have that Wikipedia article. The next five columns contain title of five most used categories with the corresponding number of occurrence. The first row shows that article "Europe" appeared in 106 languages, in 102 languages it belonged to the "Category:Europe", in 48 languages it belonged to the "Category:Continent" and in 12 languages it belonged to the "Category:Geography". In this case the probabilities that article "Europe" belongs to the categories "Europe", "Continents" and "Geography" are 0.962, 0.452 and 0.113, respectively.

Europe	106	europe:102	continents:48	geograhy:12	peninsulas of asia:4	cultural concepts:1
Geography	104	geography:99	earth sciences:15	category:social_sciences:13	academic disciplines:9	
Internet	102	internet:94	american inventions:10	category:digital_technology:8	media technology:5	1969 in the united states:4
Tehran	96	cities_in_iran:51	capitals in asia:44	category:tehran:37	populated places along the silk road:12	iranian provincial capitals:11
Copper	94	chemical_elements:70	copper:33	category:transition_metals:25	category:dietary_minerals:9	electrical conductors:8

Figure D.1: A part of the lookup data used by Sunflower. The system builds this data for every named entity article in the Wikipedia based on different languages. In this example, as shown in the first row, the article "Europe" appeared in 106 languages, in 102 languages it belonged to the "Category:Europe", in 48 languages it belonged to the "Category:Continent" and in 12 languages it belonged to the "Category:Geography".

The input of Sunflower is a Wikipedia article and its output is a weighted and directed graph of categories. In this graph, direction of edges are from subcategory to the parent category. The weight of edges are calculated by Eq. D.1. To calculate the weight of each node, the parent's weight is multiplied by edge's weight. If a node has more than one parent, we select the highest score as its weight. Considering the graph in Fig. D.2, the weights of nodes are computed as following:

- weight(B) $= 1 \times 0.63 = 0.63$
- weight(C) $= 1 \times 0.0.28 = 0.28$
- weight(D) $= 0.63 \times 0.77 = 0.485$

- weight(E) = $max(0.63 \times 0.4, 0.485 \times 0.9) = 0.436$

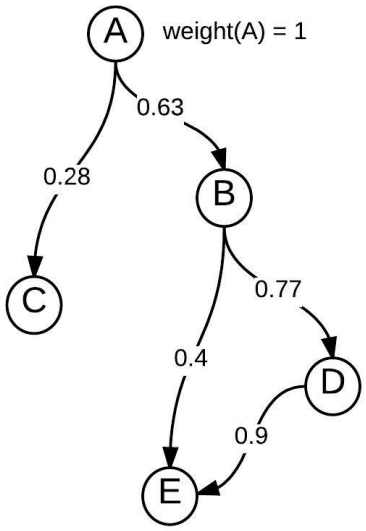

Figure D.2: An example output graph of Sunflower. This graph is weighted and directed. The direction is from the subcategory to the parent and the weight of a node is computed by multiplying weight of parent node by the weight of the edge between them.

To specify the generality level or the minimum relatedness of category to the initial concept, Sunflower provides two configurable parameters of depth and width. The former is used to define the maximum level of nodes in the output graph and the latter determine the highest number of branches in each level of the graph.

As mentioned Sunflower was designed for the named entities, while we needed to work with all concepts of Wikipedia. Thus, we extracted the required category information for all articles from Wikipedia dump files and extended the Sunflower lookup table to support all Wikipedia articles.

Contribution of people to Wikipedia varies significantly in different languages. The top five languages based on number of articles are English, Swedish, German, Dutch and German[1] as of November 2015 where English has significantly more articles than any other languages. In order to consider the importance of English as it seems to be the most active language in Wikipedia, we decided to change the calculation of

[1]https://stats.wikimedia.org/EN/Sitemap.htm

lookup table. We count the number occurrences of a category in different languages for an article and store the top most used categories only if that category exist in the English article up to maximum of 5 categories. This way only categories that exist in the English version will be considered as the direct category of that article. However, we still use the category graph from different languages and calculate the probabilities as mentioned before.

Finally we provide a web service to generalize a Wikipedia concept and build its category graph. The system supports all articles of Wikipedia. It receives a Wikipedia article as input and two parameters as the depth and width of the output graph. The output provided by this service is the information to build the category graph of input concept that contains nodes (categories), paths and a relatedness score for every node to show similarity of that node to input. In our system we only use nodes of the graph and their associated relatedness score.

Appendix E

Wikipedia Miner

Wikipedia as a huge source of knowledge with articles that cover various subjects in different languages has been popular among researchers. The content of Wikipedia is available to download and contains everything from articles, categories, structure, links between languages, in-links, out-links and everything else that exists in the Wikipedia website. However, working with this amount of data needs a lot of resources and effort.

Wikipedia Miner [Milne and Witten, 2013] is a system developed by University of Waikato and provides services to make working with Wikipedia easier and more feasible. Wikipedia Miner offers a variety of tasks that could be done using Wikipedia. In order to do that, Wikipedia Miner stores content and structure of Wikipedia and provides a Java API to access different services.

Two important services we have used in this thesis are called annotation and comparison. The annotation service is able to wikify documents. It receives a document as input and detect important terms in it. To detect these terms first it calculate the prior link probability for all article titles, redirects and link anchors of Wikipedia. This probability is computed based on the usage statistics in Wikipedia. For instance, in 76% of the times the term dog is linked to the article pet [Milne and Witten, 2013]. The term detector then extract all terms from the text of input document with the link prior probability higher than a specific threshold. These labels are then disambiguated and for each label a single Wikipedia article is returned with a score that shows the relevance of that article to the document. An example of a wikified document belong to the abstract of a paper is shown in Fig E.1.

The next service we used is called comparison. This service receives two terms and evaluate the semantic relatedness of them. In case that input terms are both Wikipedia articles, it computes a set of features based on the in-links and out-links of associated articles e.g., normalized link distance which is modeled using Google

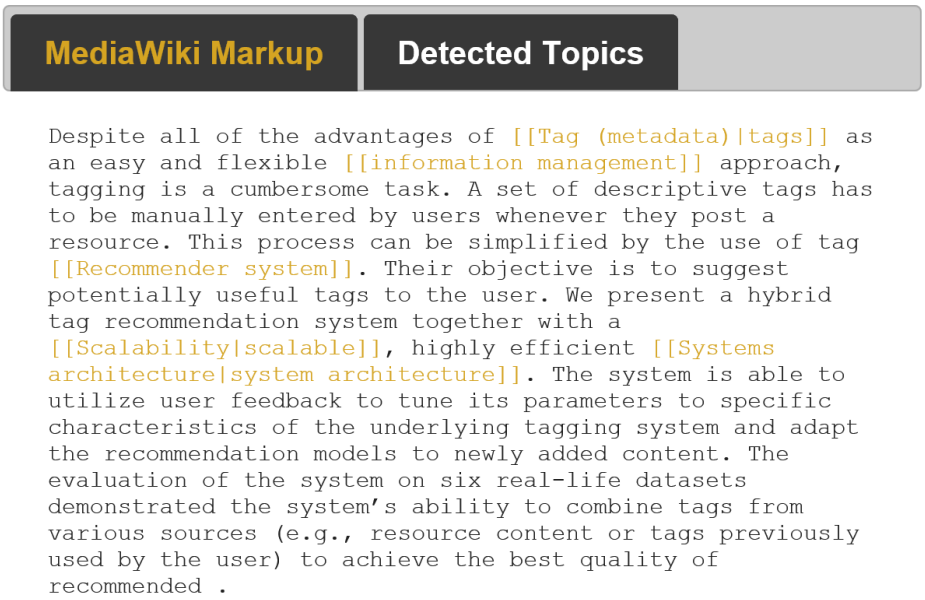

Figure E.1: A wikified example of a paper abstract using Wikipedia Miner system. The selected keyterms are shown in the brackets [[keyterm]] and if the title of associated Wikipedia article is different from the keyterm, the title is written after | in the bracket otherwise nothing is shown as the article title. Hovering over each keyterm shows its relatedness score.

distance and link vector similarity which is modeled using the TF-IDF vectors. Based on these features a model is trained and its performance on automatic calculation of input terms' relatedness is tested.

Appendix F

Weka Parameters

In this section we report the configuration and parameters we used to perform classification using Weka. We use the Weka documentation to describe each parameter. We use the version 3.6 and anything that is not reported here has the default configuration.

F.1 Decision Tree

The "confidenceFactor" is the threshold that is used for pruning and equals 0.25 and "minNumObj" is the minimum number of instances per leaf which is equal to 2.

F.2 Random Forest

"MaxDepth" is equal to zero which allows the unlimited depth for the tree and "numTrees" equals 100 which builds 100 trees.

F.3 SVM

For the SVM we are using the "LibSVM" library which is an integrated plug-in in Weka. We use a linear kernel and the "cost" parameter equals 10. The "coef0" is set to 10 which indicates the coefficient to use and "eps" is equal to 0.1 that is the termination tolerance.

F.4 Multilayer Perceptron

For this task we use "multilayerPerceptron" wihch is a multi-class classifier. The "hiddenlayers" is set to "a" which is a default value and set the number of hidden layers to (classes+attributes)/2. The learning rate is 0.3 and momentum is 0.2. The "trainingtime" equals 500 which is the number of epochs used in training.