Keyphrase extraction and its applications to digital libraries

by

Krutarth Indubhai Patel

B.Tech., Dharmsinh Desai University, India, 2014

M.S., University of North Texas, USA, 2017

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Abstract

Scholarly digital libraries provide access to scientific publications and comprise useful resources for researchers. Moreover, they are very useful in many applications such as document and citation recommendation, expert search, scientific paper summarization, collaborator recommendation, topic classification, and keyphrase extraction. Despite the advancements in search engine features, ranking methods, technologies, and the availability of programmable APIs, current-day open-access digital libraries still rely on crawl-based approaches for acquiring their underlying document collections. Furthermore, keyphrases associated with research papers provide an effective way to find useful information in the large and growing scholarly digital collections. Keyphrases are useful in many applications such as document indexing and summarization, topic tracking, contextual advertising, and opinion mining. However, keyphrases are not always provided with the papers, but they need to be extracted from their content. A growing number of scholarly digital libraries, museums, and archives around the world are embracing web archiving as a mechanism to collect born-digital material made available via the web. To create the specialized collection from the Web archived data, there is a substantial need for automatic approaches that can distinguish the documents of interest for a collection.

In this dissertation, we first explore keyphrase extraction as a supervised task and formulated as sequence labeling and utilize the power of Conditional Random Fields in capturing label dependencies through a transition parameter matrix consisting of the transition probabilities from one label to the neighboring label. Our proposed CRF-based supervised approach exploits word embeddings as features along with traditional, document-specific features. Our results on five datasets of research papers show that the word embeddings combined with document-specific features achieve high performance and outperform strong baselines for this task. We also propose KPRank, an unsupervised graph-based algorithm for

keyphrase extraction that exploits both positional information and contextual word embeddings into a biased PageRank. Our experimental results on five benchmark datasets show that KPRank that uses contextual word embeddings with additional position signal outperforms previous approaches and strong baselines for this task. Furthermore, we investigate and contrast three supervised keyphrase extraction models to explore their deployment in CiteSeerX digital library for extracting high-quality keyphrases.

Further, we propose a novel search-driven framework for acquiring documents for such scientific portals. Within our framework, publicly-available research paper titles and author names are used as queries to a Web search engine. We were able to obtain $\approx 267,000$ unique research papers through our fully-automated framework using $\approx 76,000$ queries, resulting in almost $200,000$ more papers than the number of queries. Furthermore, We propose a novel search-driven approach to build and maintain a large collection of homepages that can be used as seed URLs in any digital library including CiteSeerX to crawl scientific documents. We use *Self-Training* in order to reduce the labeling effort and to utilize the unlabeled data to train the efficient researcher homepage classifier. Our experiments on a large-scale dataset highlight the effectiveness of our approach, and position Web search as an effective method for acquiring authors' homepages.

Finally, we explore different learning models and feature representations to determine the best-performing ones for identifying the documents of interest from the web archived data. Specifically, we study both machine learning and deep learning models and "bag of words" (BoW) features extracted from the entire document or from specific portions of the document, as well as structural features that capture the structure of documents. Moreover, we explore dynamic fusion models to find, on the fly, the model or combination of models that perform best on a variety of document types. We proposed two dynamic classifier selection algorithms: Dynamic Classifier Selection for Document Classification (or DCSDC), and Dynamic Decision level Fusion for Document Classification (or DDFC). Our experimental results show that the approach that fuses different models outperforms individual models and other ensemble methods on all three datasets.

Keyphrase extraction and its applications to digital libraries

by

Krutarth Indubhai Patel

B.Tech., Dharmsinh Desai University, India, 2014

M.S., University of North Texas, USA, 2017

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Co-Major Professor
Dr. Cornelia Caragea

Approved by:

Co-Major Professor
Dr. Doina Caragea

Copyright

Abstract

Scholarly digital libraries provide access to scientific publications and comprise useful resources for researchers. Moreover, they are very useful in many applications such as document and citation recommendation, expert search, scientific paper summarization, collaborator recommendation, topic classification, and keyphrase extraction. Despite the advancements in search engine features, ranking methods, technologies, and the availability of programmable APIs, current-day open-access digital libraries still rely on crawl-based approaches for acquiring their underlying document collections. Furthermore, keyphrases associated with research papers provide an effective way to find useful information in the large and growing scholarly digital collections. Keyphrases are useful in many applications such as document indexing and summarization, topic tracking, contextual advertising, and opinion mining. However, keyphrases are not always provided with the papers, but they need to be extracted from their content. A growing number of scholarly digital libraries, museums, and archives around the world are embracing web archiving as a mechanism to collect born-digital material made available via the web. To create the specialized collection from the Web archived data, there is a substantial need for automatic approaches that can distinguish the documents of interest for a collection.

In this dissertation, we first explore keyphrase extraction as a supervised task and formulated as sequence labeling and utilize the power of Conditional Random Fields in capturing label dependencies through a transition parameter matrix consisting of the transition probabilities from one label to the neighboring label. Our proposed CRF-based supervised approach exploits word embeddings as features along with traditional, document-specific features. Our results on five datasets of research papers show that the word embeddings combined with document-specific features achieve high performance and outperform strong baselines for this task. We also propose KPRank, an unsupervised graph-based algorithm for

keyphrase extraction that exploits both positional information and contextual word embeddings into a biased PageRank. Our experimental results on five benchmark datasets show that KPRank that uses contextual word embeddings with additional position signal outperforms previous approaches and strong baselines for this task. Furthermore, we investigate and contrast three supervised keyphrase extraction models to explore their deployment in CiteSeerX digital library for extracting high-quality keyphrases.

Further, we propose a novel search-driven framework for acquiring documents for such scientific portals. Within our framework, publicly-available research paper titles and author names are used as queries to a Web search engine. We were able to obtain $\approx 267,000$ unique research papers through our fully-automated framework using $\approx 76,000$ queries, resulting in almost $200,000$ more papers than the number of queries. Furthermore, We propose a novel search-driven approach to build and maintain a large collection of homepages that can be used as seed URLs in any digital library including CiteSeerX to crawl scientific documents. We use *Self-Training* in order to reduce the labeling effort and to utilize the unlabeled data to train the efficient researcher homepage classifier. Our experiments on a large-scale dataset highlight the effectiveness of our approach, and position Web search as an effective method for acquiring authors' homepages.

Finally, we explore different learning models and feature representations to determine the best-performing ones for identifying the documents of interest from the web archived data. Specifically, we study both machine learning and deep learning models and "bag of words" (BoW) features extracted from the entire document or from specific portions of the document, as well as structural features that capture the structure of documents. Moreover, we explore dynamic fusion models to find, on the fly, the model or combination of models that perform best on a variety of document types. We proposed two dynamic classifier selection algorithms: Dynamic Classifier Selection for Document Classification (or DCSDC), and Dynamic Decision level Fusion for Document Classification (or DDFC). Our experimental results show that the approach that fuses different models outperforms individual models and other ensemble methods on all three datasets.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to appreciate all of the people who have supported my research and this dissertation. I begin by thanking my advisor, Dr. Cornelia Caragea, for her patience, guidance and support during my graduate studies. I deeply admire Dr. Cornelia Caragea for her persistence in the pursuit of long-term visions. I would like to thank all other members of my dissertation committee, Dr. Doina Caragea, Dr. Daniel Andresen, and Dr. Caterina Scoglio and the outside chair Dr. Jaebeom Suh. Their feedback was valuable and helped me improve my dissertation.

I am very grateful to my colleagues from Kansas State University, University of North Texas, and Column for their support during my graduate studies. I would like to thank the department of Computer Science at Kansas State University for giving me an admission and financial support for my graduate study.

Finally, my deepest appreciation to my parents, Indubhai Patel and Kalpanaben Patel, my brother Vishrut Patel, my sister Shruti Patel, my brother-in-law Jignesh Patel, and my grand parents, Babubhai Patel and Savitaben Patel for their support, encouragement, prayers, and being there for me in every situation. I would also like to thank all of my friends from Dharmsinh Desai University, University of North Texas, and Kansas State University.

I dedicate this dissertation to my parents for their unconditional love and support throughout my entire life.

Chapter 1

Introduction

In this chapter, we discuss the background and motivation of our study on keyphrase extraction and its applications to digital libraries.

1.1 Background

Scientific portals such as Google Scholar, Semantic Scholar, ACL Anthology, CiteSeer^{x}, and ArnetMiner, provide access to scholarly publications and comprise indispensable resources for researchers who search for literature on specific subject topics. Moreover, many applications such as document and citation recommendation[1–3], expert search[4;5], topic classification[6;7], and keyphrase extraction and generation[8–11], involve Web-scale analysis of up-to-date research collections. Moreover, open access scientific portals usually collect the documents from online repositories as well as maintains a list of seed URLs for crawling the documents.

Keyphrases associated with research papers provide an effective way to find useful information in the large and growing scholarly digital collections. Keyphrase extraction is the task of automatically extracting a small set of descriptive words or phrases that can accurately summarize the topics discussed in a document[12;13]. Keyphrases are useful in many applications such as document indexing, classification, clustering, recommendation, and summarization[14–17], contextual advertising[18], and opinion mining[19]. Most of the previ-

ous approaches to keyphrase extraction are either supervised or unsupervised. Due to their high importance, many approaches to keyphrase extraction have been proposed in the literature. While supervised approaches perform generally better[20], the unsupervised ones have the advantage that they do not require large human-annotated corpora for training reliable models. Most of the keyphrase extraction approaches work in two steps. First, a set of candidate words or phrases are formed using certain part-of-speech (POS) tags (e.g., nouns and adjectives) and patterns (e.g., at least one noun possibly preceded by adjectives)[21]. Second, the candidate words or phrases are ranked based on the aggregated "informativeness" scores of the individual words comprising a phrase[22;23] in unsupervised approaches, or are classified as keyphrases or non-keyphrases based on a set of linguistic and statistical features such as *tf-idf*, POS tags, and the relative position of phrases in documents[21;24] in supervised approaches.

A growing number of research libraries, museums, and archives around the world are embracing web archiving as a mechanism to collect born-digital material made available via the web. The Web archived data can also provide access to the documents of different types and domains. The Web archived data usually contains high-quality documents that are very useful for creating specialized collections of documents. To create such collections, there is a substantial need for automatic approaches that can distinguish the documents of interest for a collection out of the large collections (of millions in size) from Web Archiving institutions. The amount of data that these web archiving initiatives generate is typically at levels that dwarf traditional digital library collections. As an example, in a recent impromptu analysis, Jefferson Bailey of the Internet Archive noted that there were 1.6 Billion PDF files in the Global Wayback Machine[25]. If just 1% of these PDFs are of interest for collection organizations, that would result in a collection larger than the 15 million volumes in HathiTrust[26].

1.2 Motivation and Contributions

Open access scientific portals usually collect the documents from online repositories as well as maintains a list of seed URLs for crawling the documents. However, given the dynamic nature of the Web, maintaining comprehensive, up-to-date collections of seed URLs is very challenging task. For automatically augmenting the document collections, we propose a novel framework based on Web search. To motivate our framework, we recall how a Web user typically searches for research papers or authors. As with regular document search, a user typically issues Web search queries comprising of representative keywords or paper titles for finding publications on a topic. Similarly, if the author is known, a "navigational query" [27] may be employed to locate the homepage where the paper is likely to be hosted. For maintaining the accurate list of researchers' homepages, one approach would be to crawl academic websites (e.g., from a university domain) and use a machine learning classifier to predict whether a website accessed during the crawl is an author homepage or not [28]. However, this approach: (1) is still inefficient (e.g., in terms of bandwidth and storage resources) since only a small fraction of the websites hosted in an academic domain are author homepages (with many websites corresponding to departments, courses, groups, etc), and (2) misses homepages from research industry labs, which do not belong to the academic domain (e.g., ∼51% of the homepages in our dataset are not from the .edu domain). An alternative, more efficient approach, is to use a broader Web search for author discovery together with an accurate homepage classifier.

For the keyphrase extraction, in the supervised setting, researchers have started to address keyphrase extraction as a sequence labeling task [29–31]. For example, Gollapalli et al. [29] formulated keyphrase extraction as sequence labeling and showed on several datasets of research papers that using Conditional Random Fields (CRFs) can improve the performance over previous supervised and unsupervised models for this task. The authors used word features such as "WordIsCapitalized," "WordIsStopword," NP-chunking and POS tags, and "WordIsInTitle," as well as their combinations, e.g., "WordIsInTitle and Word POS tag=noun." However, this approach does not capture the semantics of words in context

that are often hidden in text. We posit that incorporating word semantics in context in a CRF model has the potential to further improve the performance of keyphrase extraction from research papers. In the unsupervised setting, more recently, Mahata et al.[32] proposed a theme-weighted biased PageRank, called Key2Vec, for keyphrase extraction. In Key2Vec, a theme-vector is computed by averaging the embeddings of words and phrases from the title of a scientific document to capture its theme and the PageRank is biased based on the similarity of candidate words or phrases to the computed theme vector. However, this model is oblivious to the position of words in a scientific document, in which more important words appear not only frequently, but also close to the beginning of the document[33].

While the number of Web Archiving institutions increases, the technologies needed to provide access to these large collections have not improved significantly over the years. The use of full-text search has increased in many web archives around the world, but often provides an experience that is challenging for users because of the vast amount of content and the limitations of strictly text-based searches for these large heterogeneous collections of content. Our research is aimed at understanding how well machine learning and deep learning models can be employed to provide assistance to collection maintainers who are seeking to classify the PDF documents from their web archives into being within scope for a given collection or collection policy or out of scope.

The main purpose of this dissertation

In this dissertation we divided this research into three parts and they are as following.

1. Keyphrase Extraction from Scientific Documents
2. Applications of Keyphrases/Keywords
    - Scientific Documents Discovery
    - Researchers' Homapages Discovery
3. Document Classification in Web Archiving Collections

1.2.1 Keyphrase Extraction from Scientific Documents

For the keyphrase extraction task, we proposed a supervised CRF based approach that exploits un contextual word embeddings along with document specific features. Furthermore, we proposed an unsupervised keyphrase extraction approach, KPRank, that utilizes positional information of the words along with the contextual word embeddings for computing the biased pagerank score to rank candidate phrases. Furthermore, investigate and contrast three supervised keyphrase extraction models to explore their deployment in CiteSeerX digital library. In summary, our contributions are as follows:

- We propose to incorporate word semantics in CRF models for keyphrase extraction through the use of word embeddings. We study the sensitivity of CRFs based on word embedding types, i.e., those pre-trained on Google News as well as those trained on a large collection of ACM research papers. As part of our contributions, we will make available the IDs of our ACM dataset and the word embeddings.[1]

- We experimentally show that the CRF models that use word embeddings in addition to features extracted from the document itself outperform strong baselines and other previous approaches for keyphrase extraction.

- We propose KPRank, an unsupervised graph-based algorithm that exploits both the position of words in a document and the contextual word embeddings for computing a biased PageRank score for ranking candidate phrases

- We show empirically that infusing position information into our biased KPRank model yields better performance compared with its counterpart that does not use the position information. In addition, KPRank with contextual SciBERT embeddings performs better than FastText-based KPRank. Finally, we show that KPRank outperforms many previous unsupervised models.

- We review keyphrase extraction in scholarly digital libraries, using CiteSeerX as a

[1]The code and data are available upon request.

case study. Moreover, we show the development and deployment requirements of the keyphrase extraction models and the maintenance requirements.

1.2.2 Applications of Keyphrases/Keywords

Keyphrases or keywords are very useful to formulate queries that can retrieve topically-related articles from the Web. We propose two search driven approaches for acquiring scientific documents and maintaining a large collection of researcher homepages. Moreover, we also designed a traditional machine learning based researcher homepage ranking approach and a convolutional neural network based researcher homepage classifier used in our scientific document acquisition framework and researcher homepage discovery framework, respectively. Our contributions are as follows:

- We propose a novel integrated framework based on search-driven methods to automatically acquire research documents for scientific collections. To our knowledge, we are the first to use "Web Search" based on author names to obtain seed URLs for initiating crawls in an open-access digital library. Moreover, we design a traditional machine learning based novel homepage identification module and adapt existing research on academic document classification, which are crucial components of our framework. We show experimentally that our homepage identification module and the research paper classifier substantially outperform strong baselines.

- To automatically acquire research documents, we perform a large-scale, first-of-its-kind experiment using $43,496$ research paper titles and $32,816$ author names from Computer and Information Sciences. We compare our framework with two baselines, a breadth-first search crawler and, to the extent possible, the Microsoft Academic. We discuss that our framework does not substitute these systems, but rather they very well complement each other. We compare our framework with two baselines, a breadth-first search crawler and, to the extent possible, the Microsoft Academic. We discuss that our framework does not substitute these systems, but rather they very well complement

each other. As part of our contributions, we will make all the constructed datasets available.

- We propose a search-driven homepage finding approach that uses author names and paper titles to find researcher homepages. To our knowledge, we are the first to use *"paper titles"* as queries to discover researcher homepages. Furthermore, we explore Convolutional Neural Networks (CNNs) for author homepage identification,[2] which is a crucial component in our approach. We conduct a thorough evaluation of the CNN models trained on both URLs and page content, and show significant improvements in performance over baselines and prior works. Furthermore, we show that self training can improve the performance of the classifier with the small amount of labeled data along with the unlabeled data.

- To discover researcher homepages, we perform a large-scale experiment using author names and paper titles from Computer Science as queries, and show the effectiveness of our approach in discovering a large number of homepages. Finally, as part of our contributions, all resulting datasets for author homepage identification and homepage discovery will be made available to further research in this area. We show the development and deployment requirements of our proposed approach in CiteSeerX and the maintenance requirements.

1.2.3 Document Classification in Web Archiving Collections

We explore different learning models and feature representations to determine the best performing ones for identifying the documents of interest from the web archived data. Moreover, we explore dynamic fusion models to find, on the fly, the model or combination of models that performs best on a variety of document types. We focus our evaluation on three datasets that we created from three different Web archives. In summary, we make the following contributions:

[2]We use author homepage classification or identification interchangeably.

- We built three datasets from three different web archives collected by the UNT libraries, each covering different domains: UNT.edu, Texas.gov, and USDA.gov. Each dataset contains the PDF document along with the label indicating whether a document is in scope of a collection or not. We will make these datasets available to further research in this area.

- We show that BoW classifiers that use only some portion of the documents outperform BoW classifiers that use full text from the entire content of a document, the structural features based classifiers, and the CNN classifier. We also show that feature selection using information gain improves the performance of the BoW classifiers and structural features based classifiers, and present a discussion on the most informative features for each collection.

- We propose a dynamic classifier selection for document classification (DCSDC) to dynamically select an appropriate classifier to predict the probability of a target document as being in scope of a collection or not. To dynamically select the classifiers, we consider textual similarity along with the structural aspects of the documents. We show that DCSDC outperforms all the individual feature set models (base classifiers) and other strong baselines.

- We propose a dynamic decision-level fusion for document classification (DDFC) that derives competence features from neighborhood documents and learns a classifier to assign a competence score for each base classifier (BoW, Str, and CNN) in order to fuse them and to predict the probability of a target document as being in scope of a collection or not. To derive the competence features, we consider textual similarity along with the structural aspects of the documents. We show that DDFC outperforms all the individual feature set models (base classifiers) and other strong baselines including DCSDC.

1.3 Dissertation Outline and Published Work

In what follows, we provide a brief description of the chapters in the dissertation. Each chapter corresponds to multiple papers. The research work of the dissertation has been published either in conference proceedings, or will be submitted to a conference proceeding. Our goal is to study keyphrase extraction and its applications to digital libraries. Precisely, we propose CRF-based supervised and an unsupervised keyphrase extraction algorithm named KPRank. We also explored integration of different keyphrase extraction models in to CiteSeerX. Moreover, we propose two search driven approaches for acquiring scientific documents, and maintaining a large collection of researcher homepages. We use research paper titles (keywords) and author names as queries in our frameworks. Furthermore, we explored different machine learning and deep learning models for identifying documents in-scope of a collection from Web archives. We explore dynamic fusion models to find, on the fly, the model or combination of models that performs best on a variety of document types. We proposed two dynamic classifier selection algorithms: Dynamic Classifier Selection for Document Classification (or DCSDC), and Dynamic Decision level Fusion for Document Classification (or DDFC).

This dissertation is structured as follows:

Chapter 2: In this chapter, we present our proposed work on keyphrase extraction tasks that is briefly described in Section 1.2.1. The proposed CRF-based supervised keyphrase extraction approach is published in the 10th International Conference on Knowledge Capture (K-Cap 2019)[10]. Moreover, the proposed KPRank, an unsupervised keyphrase extraction algorithm, is accepted at the 16th Conference of the European Chapter of the Association for Computational Linguistics (ECAL 2021). And, the explorarion of the deployment of keyphrase extrations models in CiteSeerX is published in the 2020 International Conference on Web Services (ICWS 2020)[34].

Chapter 3: In this chapter, we present our proposed work on the applications of kephrases/keywords for acquiring scientific documents and maintaining a large collection of researcher homepages as briefly described in Section 1.2.2. The proposed framework for augmenting the document collections in the scientific digital library is published in the First

Workshop on Scholarly Document Processing (SDP@EMNLP 2020)[35]. And, the proposed framework for maintaining the accurate list of researchers' homepages is published in the Thirty-Third Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-21)[36].

Chapter 4: In this chapter, we present our work on the document classification task in the Web archiving collections briefly explained in Section 1.2.3. The exploration of different base classifiers for identifying documents in-scope of a collection from Web archives is published in the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2020)[37]. Our proposed dynamic classifier selection approach (DCSDC) is published in the 12th Language Resources and Evaluation Conference (LREC 2020)[38]. Moreover, our competency learning-based dynamic classifier selection approach (DDFC) will be submitted to the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2021).

Chapter 5: We summarize and conclude the dissertation. We also provide a summary of contributions and directions for future research.

Chapter 2

Keyphrase Extraction from Scientific Documents

Keyphrases associated with research papers provide an effective way to find useful information in the large and growing scholarly digital collections. However, keyphrases are not always provided with the papers, but they need to be extracted from their content. In this chapter, we explore keyphrase extraction as a supervised task and formulated as sequence labeling and utilize the power of Conditional Random Fields in capturing label dependencies through a transition parameter matrix consisting of the transition probabilities from one label to the neighboring label. We aim at identifying the features that, by themselves or in combination with others, perform well in extracting the descriptive keyphrases for a paper. Specifically, we explore word embeddings as features along with traditional, document-specific features for keyphrase extraction. Our results on five datasets of research papers show that the word embeddings combined with document specific features achieve high performance and outperform strong baselines for this task. Moreover, we also propose KPRank, an unsupervised graph-based algorithm for keyphrase extraction that exploits both positional information and contextual word embeddings into a biased PageRank. Our experimental results on five benchmark datasets show that KPRank that uses contextual word embeddings with additional position signal outperforms previous approaches and strong baselines for this task.

Scholarly digital libraries provide access to scientific publications and comprise useful resources for researchers who search for literature on specific subject areas. CiteSeerX is an example of such a digital library search engine that provides access to more than 10 million academic documents and has nearly one million users and three million hits per day. Artificial Intelligence (AI) technologies are used in many components of CiteSeerX including Web crawling, document ingestion, and metadata extraction. CiteSeerX also uses an unsupervised algorithm called noun phrase chunking (NP-Chunking) to extract keyphrases out of documents. However, often NP-Chunking extracts many unimportant noun phrases. Thus, we investigate and contrast three supervised keyphrase extraction models to explore their deployment in CiteSeerX for extracting high quality keyphrases. To perform user evaluations on the keyphrases predicted by different models, we integrate a voting interface into CiteSeerX. We show the development and deployment of the keyphrase extraction models and the maintenance requirements.

2.1 Introduction

Keyphrase extraction is the task of automatically extracting a small set of meaningful words or phrases that can accurately summarize the topics discussed in a document[12]. Keyphrases therefore provide a high-level topic description of a document, can allow for *efficient* data organization and information processing, and have a high impact on document understanding and reading comprehension. Additionally, keyphrases associated with a document are often useful in many applications such as document indexing, classification, clustering, recommendation, and summarization[14–17], contextual advertising[18], and opinion mining[19]. Due to their high importance, many approaches to keyphrase extraction have been proposed in the literature. Most of these approaches are either supervised or unsupervised and work in two steps. First, a set of candidate words or phrases are formed using certain part-of-speech (POS) tags (e.g., nouns and adjectives) and patterns (e.g., at least one noun possibly preceded by adjectives)[21]. Second, the candidate words or phrases are ranked based on the aggregated "informativeness" scores of the individual words comprising a phrase[22;23] in un-

supervised approaches, or are classified as keyphrases or non-keyphrases based on a set of linguistic and statistical features such as *tf-idf*, POS tags, and the relative position of phrases in documents[21;24] in supervised approaches.

In the supervised setting, more recently, researchers started to address keyphrase extraction as a sequence labeling task[29–31]. For example, Gollapalli et al.[29] formulated keyphrase extraction as sequence labeling and showed on several datasets of research papers that using Conditional Random Fields (CRFs) can improve the performance over previous supervised and unsupervised models for this task. The authors used word features such as "WordIsCapitalized," "WordIsStopword," NP-chunking and POS tags, and "WordIsInTitle," as well as their combinations, e.g., "WordIsInTitle and Word POS tag=noun." However, this approach does not capture the semantics of words in context that are often hidden in text. We posit that incorporating word semantics in context in a CRF model has the potential to further improve the performance of keyphrase extraction from research papers.

One way to capture word semantics is to use word embeddings or the distributed vector representations of words, trained on very large collections of documents in an unsupervised fashion[39]. Along this line, Turian et al.[40] showed on two NLP tasks, chunking and named entity recognition, that using word embeddings in existing supervised systems is a simple and general method to improve their performance. Marujo et al.[41] and Mahata et al.[32] showed promising results on keyword extraction by incorporating word embeddings into supervised and unsupervised models. However, unlike these works, we investigate the discriminative power of word embeddings in sequence labeling based models. Specifically, we address keyphrase extraction as sequence labeling using CRF models and explore word embeddings as features, by themselves or in combination with other statistical and linguistic features, to determine their discriminative power in correctly extracting the descriptive keyphrases for a research paper.

In the unsupervised setting, more recently, Mahata et al.[32] proposed a theme-weighted biased PageRank, called Key2Vec, for keyphrase extraction. In Key2Vec, a theme-vector is computed by averaging the embeddings of words and phrases from the title of a scientific document to capture its theme and the PageRank is biased based on the similarity of can-

didate words or phrases to the computed theme vector. However, this model is oblivious to the position of words in a scientific document, in which more important words appear not only frequently, but also close to the beginning of the document[33]. Inspired by the Transformer models[42] that infuse positional information into the word embeddings to produce embeddings with time signal, we propose an extension of Key2Vec that incorporates words' positions into a biased PageRank. Moreover, different from Mahata et al.[32], who used non-contextual FastText embeddings[43], we propose to integrate SciBERT contextual embeddings[44] into our biased PageRank extension.

Due to the high importance of keyphrases, several online digital libraries such as the ACM Digital Library have started to impose the requirement for author-supplied keyphrases. Specifically, these libraries require authors to provide keyphrases that best describe their papers. However, keyphrases have not been integrated into all sharing mechanisms. For example, the AAAI digital library (http://www.aaai.org/) does not provide keyphrases associated with the papers published in the AAAI conferences. In an effort to understand the coverage of papers with author-supplied keyphrases in open access scholarly digital libraries, we performed the following analysis: we randomly sampled $2,000$ papers from CiteSeerX, and manually inspected each paper to determine whether a paper contains author-supplied keyphrases and if the paper was published by ACM. Note that in most of the ACM conference proceeding templates, the authors need to provide keyphrases (keywords) after the "Abstract" section. For completeness, the ACM templates from years 1998, 2010, 2015, and 2017 were adopted for visual inspection. Out of our $2,000$ sample, only 31 (1.5%) papers were written using ACM templates and only 769 papers (38%) contain author-supplied keyphrases. Out of 31 papers written using ACM templates, 25 contain author-supplied keyphrases. The fact that around 62% of papers sampled do not have author-supplied keyphrases indicates that automatic keyphrase extraction is needed for scholarly digital libraries. The CiteSeerX website currently displays keyphrases extracted using an unsupervised phrase chunking method[45]. Thus, we review keyphrase extraction in scholarly digital libraries, using CiteSeerX as a case study. . We investigate the impact of displaying keyphrases on promoting paper downloading by analyzing search engine access logs in three years from 2016 to 2018. Then, we

interrogate the quality of several supervised keyphrase extraction models to explore their deployment in CiteSeerX and perform a large scale keyphrase extraction - first of its kind for this task. Moreover, to get user evaluations on the predicted keyphrases on a large scale, we implement and integrate a voting interface, which is widely used in social networks and multimedia websites, such as Facebook and YouTube. We show the development and deployment requirements of the keyphrase extraction models and the maintenance requirements.

Our contributions are as follows:

- We propose to incorporate word semantics in CRF models for keyphrase extraction through the use of word embeddings. We study the sensitivity of CRFs based on word embedding types, i.e., those pre-trained on Google News as well as those trained on a large collection of ACM research papers. As part of our contributions, we will make available the IDs of our ACM dataset and the word embeddings.[1]

- We experimentally show that the CRF models that use word embeddings in addition to features extracted from the document itself outperform strong baselines and other previous approaches for keyphrase extraction.

- We propose KPRank, an unsupervised graph-based algorithm that exploits both the position of words in a document and the contextual word embeddings for computing a biased PageRank score for ranking candidate phrases

- We show empirically that infusing position information into our biased KPRank model yields better performance compared with its counterpart that does not use the position information. In addition, KPRank with contextual SciBERT embeddings performs better than FastText-based KPRank. Finally, we show that KPRank outperforms many previous unsupervised models.

- We review keyphrase extraction in scholarly digital libraries, using CiteSeerX as a case study. Moreover, we show the development and deployment requirements of the keyphrase extraction models and the maintenance requirements.

[1]The code and data are available upon request.

2.2 Related Work

Keyphrase extraction has been the focus of many supervised and unsupervised studies[12]. In the supervised studies, the prediction is done based on a selection of linguistic and statistical features extracted from the text of a document, e.g., a word or phrase part of speech (POS) tags, *tf-idf* scores, and position information, used in conjunction with machine learning classifiers such as Naïve Bayes and Support Vector Machines[21;24;46;47]. These features were also combined with features extracted from external sources such as WordNet and Wikipedia[48;49] or from various document neighborhoods, e.g., a document's citation network[11;50].

Unsupervised studies include phrase scoring methods based on measures such as *tf-idf* and topic proportions[51–53], graph-based ranking using centrality measures, e.g., PageRank scores[22;23;33;54;55], and keyphrase selection from topics detected using topic modeling[56;57]. Blank et al.[58] ranked keyphrases for a target paper using keyphrases from the papers that are cited by the target paper or that cite at least one paper that the target paper cites. In the unsupervised context, several extensions of PageRank have been proposed that make use of a document's citation network[59] or that bias the random walk based on the words' positions in text[33] or the words' topic distribution estimated using topic models[60]. In order to add semantic relatedness between the words in a word graph, Martinez-Romo et al.[61] used information from WordNet. The best performing SemEval 2010 system used term frequency thresholds to filter out phrases that are unlikely to be keyphrases, where the thresholds were estimated from the data[62]. The candidate phrases were ranked using *tf-idf* in conjunction with a boosting factor that was aimed at reducing the bias towards single words. Danesh et al.[63] computed an initial weight for each phrase based on a combination of the *tf-idf* score and the first position of a phrase in a document. Phrases and their initial weights were then incorporated into a graph-based algorithm which produces the final ranking of keyphrases. Adar and Datta[64] extracted keyphrases by mining abbreviations from scientific literature and built a semantic hierarchical keyphrase database. Many of the above approaches, both supervised and unsupervised, are compared and analyzed in the ACL survey on keyphrase extraction by Hasan and Ng[13]. Fan et al.[65] used a random-walk method

to extract keyphrases using word embeddings combined with the features of candidate words and edges from the word graph.

Neural networks and word embeddings have started to be incorporated into models for keyphrase extraction. For example, Wang et al.[66] investigated word embeddings to measure the relatedness between words in graph-based models. Marujo et al.[41] used word embeddings in the existing supervised MAUI system[48] to extract keywords from tweets. Mahata et al.[32] exploited word and phrase embeddings in an unsupervised topic-biased PageRank to extract keyphrases from research papers and showed improvements in performance over models that do not use embeddings. Bennani-Smires et al.[67] explored simple models for keyphrase extraction based on sentence embeddings. A Recurrent Neural Network (RNN) based approach was proposed by Zhang et al.[68] to identify keyphrases in Twitter data, using a joint-layer RNN to capture the semantic dependencies in the input sequence, but did not address the dependencies in the labels. Ray Chowdhury et al.[69] extended this joint-layer RNN to capture informal writing in tweets. Augenstein and Søgaard[70] used multi-task learning to classify keyphrase boundaries.

Inspired from work in machine translation, Meng et al.[71] focused on keyphrase generation (rather than keyphrase extraction) and addressed the task as a sequence to sequence learning problem, where the sequence of words in a document is used to generate a sequence of keyphrases. An Encoder-Decoder RNN, originally proposed by Cho et al.[72], was used to generate the keyphrase sequences. Several other works focused on keyphrase generation[73–75]. Unlike these works, we focus on keyphrase extraction and not keyphrase generation, which generates words that may or may not be present in the text. Specifically, we mine the content of documents to extract keyphrases that are present in their content, using CRF-based sequence labeling and the power of unsupervised word embeddings.

Sequence labeling models for keyphrase extraction have shown promising results in several studies[29–31]. For example, Gollapalli et al.[29] trained a CRF to extract keyphrases from scholarly documents, using features such as *tf-idf* and POS tags to predict a label for each token position in a document as being a keyphrase token (**KP**) or not (**Non-KP**). Recently, a sequence labeling framework has been explored on a variety of NLP tasks such as POS

tagging, noun phrase chunking, named entity recognition, and keyphrase extraction[76–79] that combines a Bi-LSTM network as the first layer to capture sequential text dependencies with a second CRF layer to capture label dependencies.

In our work, we explore word embeddings in CRF models as a simple and general approach for keyphrase extraction from scholarly documents, and contrast this with the more sophisticated model, Bi-LSTM-CRF. To our knowledge, in the context of keyphrase extraction from scholarly documents, we are the first to directly use word embeddings as features in CRF-based models and show improved results over previous models. Moreover, different from Mahata et al.[32], who used non-contextual FastText embeddings[43], we propose to integrate SciBERT contextual embeddings[44] into our biased PageRank extension.

2.3 Proposed Approaches and Methods Used in CiteSeerX Case Study

First, we discuss our proposed CRF-based supervised keyphrase extraction approach. Second, we discuss our proposed KPRank, an unsupervised keyphrase extraction algorithm. Last, we discuss different keyphrase extraction methods that we used in CiteSeerX case study.

2.3.1 CRF-Based Supervised Keyphrase Extraction

CRF: We formulate keyphrase extraction as sequence labeling using Conditional Random Fields[80]. CRFs combine the advantages of graphical modeling and discriminative classification and have majors advantages over other graphical models, e.g., the ability to handle a large number of rich features and the ability to avoid the label bias problem (favoring the states with less outgoing transition) by using global normalization and accounting for the entire sequence at once. In CRFs, for an input sequence $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, where each $\mathbf{x}_i$ represents the feature vector for the i^{th} word w_i in the sequence, the task is to predict a sequence of labels $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$, where y_i is assigned to w_i. In our CRF model, each

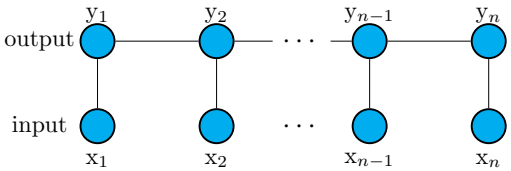


Figure 2.1: *Layers in a CRF network.*

feature vector $\mathbf{x}_i$ consists of individual or combinations of two types of features, document specific features and word embeddings, described below. Figure 2.1 shows a simple CRF network. The features that we use in our CRF models are described below.

Document Specific Features (DOC): We use six features for each word that are extracted from the target paper: a word's POS tag **(POS)**[21], term frequency-inverse document frequency **(tf-idf)** computed based on the target paper[24]; the position of the first occurrence of a word normalized by the length of the target paper (in the number of tokens) **(relative position)**[21;24]; the distance of the first occurrence of a word from the beginning of a paper **(first position)**[11]; a binary feature indicating the presence of the word in the title **(is-in-title)**[81;82]; and a binary feature indicating whether the word was capitalized **(is-capitalized)**[29]. The choice of these features was motivated by their good performance in previous models[11;21;24].

Word Embedding Features (EMB): Word embeddings are vector representations of words as dense vectors. Precisely, using word embeddings, words are expressed as dense vectors by projecting each word into a multidimensional vector space. The position of a word within the vector space or the embedding is learned from the text by considering the surrounding words from its local context, and hence, can capture the semantic relations from text. We use the components of multidimensional word embeddings of each word as its features.

Bi-LSTM-CRF: In order to build a sequence labeling model that incorporates long distance information over a sequence of input as well as information on the output sequence, we consider the Bi-LSTM-CRF network as a more sophisticated and complex model. The network architecture is shown in Figure 2.2. As shown in the figure, the first layer of the model is a Bi-LSTM network with the purpose of capturing the semantics of the input text

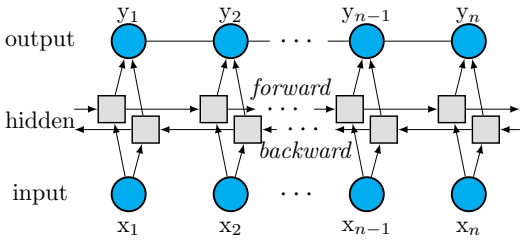


Figure 2.2: *Layers in a Bi-LSTM-CRF network.*

sequence. The output of the Bi-LSTM layer is passed to a CRF layer that produces a probability distribution over the tag sequence using the dependencies among labels of the entire sequence.

2.3.2 KPRank: An Unsupervised Keyphrase Extraction Algorithm

Here, we describe our unsupervised graph-based algorithm called KPRank, that exploits both position information of the words in a document along with contextual word embeddings for computing a biased PageRank score for each candidate word. Our approach consists of three steps: (1) candidate word selection and word graph construction; (2) word scoring by biased PageRank; and (3) candidate phrase formation.

Candidate Word Selection and Graph Construction

For a target doucment D, we first apply a part-of-speech filter[2] and select only nouns and adjectives as candidate words, consistent with previous works[22;23;59]. We build a word graph $G = (N, E)$ for D using the candidate words as nodes in G. N and E are the sets of nodes and edges, respectively. We consider an edge $(n_i, n_j) \in E$ between two nodes n_i and n_j in N if the words corresponding to these nodes appear within a window of k consecutive words in the content of D. We experimented with values of k from 1 to 10 and obtained best results with $k = 10$, which is consistent with[22]. The weight of an edge (n_i, n_j), denoted as w_{ij}, is computed based on the co-occurrence count of the two words within k consecutive words in

[2]We used Python's NLTK toolkit for POS tagging.

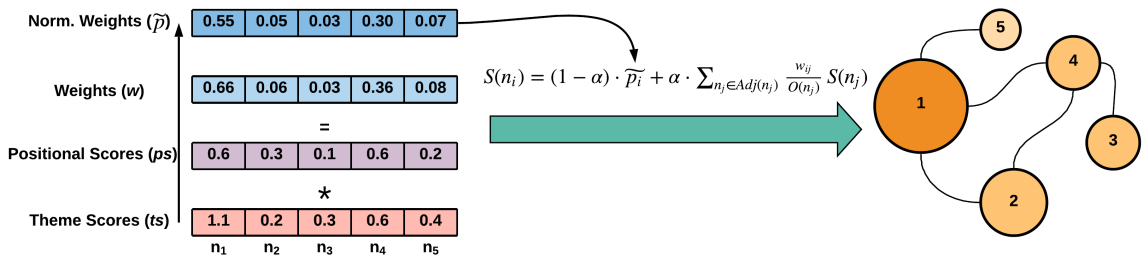


Figure 2.3: *Illustration of our approach.*

D ($k = 10$). Here, we build undirected graphs because prior work[23;60] observed that the type of graph (directed or undirected) used to represent the text does not significantly influence the performance of the keyphrase extraction task.

Biased PageRank

Preliminaries. PageRank[83] is a graph-based ranking algorithm that iteratively calculates the importance of each node in a graph through endorsements from its neighbors. For document D, we construct an undirected graph G as explained above. Initially, the score of each node in G is set to $\frac{1}{|N|}$. This score is then iteratively updated using PageRank. That is, the score s for node n_i is obtained by applying the equation:

$$s(n_i) = (1-\alpha)\widetilde{p_i} + \alpha \sum_{n_j \in Adj(n_i)} \frac{w_{ji}}{O(n_j)} s(n_j) \tag{2.1}$$

where $O(n_j) = \sum_{n_k \in Adj(n_j)} w_{jk}$ and $Adj(n_j)$ is the set of all adjacent nodes of node $n_j \in N$. $\widetilde{p_i}$ is defined below.

In order to prevent the PageRank from getting stuck in cycles or dead ends, a dumping factor α was added to Eq. (1) to allow the PageRank to randomly jump to any node in the graph ($\alpha = 0.85$). Let $\widetilde{p} = [\widetilde{p}_1, \cdots, \widetilde{p}_i, \cdots, \widetilde{p}_{|N|}]$ be the probability distribution of randomly jumping to any node in the graph. For an unbiased PageRank, this is a uniform distribution, with $\widetilde{p_i} = \frac{1}{|N|}$, for all i from 1 to $|N|$. For a biased PageRank, this probability distribution is not uniform, but rather the nodes in the graph are visited preferentially, with some nodes being visited more often than others, depending on the $\widetilde{p_i}$ value for node n_i[84]. Key2Vec is an example of (topic) biased PageRank for keyphrase extraction that computes $\widetilde{p_i}$ for node n_i using the cosine similarity between the embedding of word/phrase corresponding to

node n_i and a theme vector for the entire document, which corresponds to the aggregated word/phrase embeddings from the document's title[32]. That is, $\widetilde{p}_i$ is higher for words/phrases that are topically (semantically) more similar to the overall theme vector for the document. Next, we describe our extension KPRank of Key2Vec.

KPRank. In our proposed approach, we calculate $\widetilde{p}_i$ for node n_i using two types of scores: *theme (or topic) score* and *positional score*. We multiply both scores to assign a final weight to node n_i before running the biased PageRank algorithm. Both scores and their calculation are explained below.

To calculate the *theme score* (ts_i) for node $n_i \in N$, we first calculate a theme vector (T_D) for document D. A *theme vector* is obtained by averaging SciBERT[44] word embeddings of adjectives and nouns from D's title. The *theme score* for node n_i is calculated using the cosine similarity of the SciBERT word embedding corresponding to node n_i and T_D. The idea is to assign a higher score to a word if that word is closer to the theme (topic) of a given document. For obtaining word embeddings, for all the words with similar stemmed version (obtained with Porter stemmer), we averaged the contextualized word embeddings of a word obtained by using SciBERT. We used the title and abstract of a document as input to the SciBERT model. We also experimented with pretrained BERT[85], and found that the performance of BERT-based KPRank and SciBERT-based KPRank are very similar.

To calculate the *positional score* (ps_i) for node n_i, we consider the set P_i that contains all the positions of occurrence in the text of the word corresponding to node n_i. Then, ps_i is calculated as $ps_i = \sum_{j \in P_i} \frac{1}{j}$. For example, for a word occurring on positions 1 and 10 in the text, its ps_i score is $\frac{1}{1} + \frac{1}{10}$, whereas for a word occurring on position 100, its ps_i score is $\frac{1}{100}$. The intuition behind this weighting scheme is to give higher weight to words appearing in the beginning of a document since in scientific writing, authors tend to use keyphrases very early in the document (even from the title)[33]. Based on these considerations, the first position of a phrase/word and its relative position are also used in many supervised approaches as powerful features[10;21;86].

To calculate the weight w_i for n_i, we perform multiplication of both the *theme score* (ts_i) and the *positional score* (ps_i). The intuition is that we give preference to words that appear

near the beginning of the document and are more frequent as compared with less frequent words appearing later in document even though both words may be equally close to the theme of the document or may have similar theme score. The vector $\widetilde{p}$ is last set to the normalized weights for each node as follows:

$$\widetilde{p} = \left[\frac{w_1}{\sum_{i=1}^{|N|} w_i}, \frac{w_2}{\sum_{i=1}^{|N|} w_i}, ..., \frac{w_{|N|}}{\sum_{i=1}^{|N|} w_i} \right] \tag{2.2}$$

The biased PageRank scores for each node n_i are finally calculated by iteratively applying Eq. (1) with $\widetilde{p}$ as in Eq. (2). Figure 2.3 shows the illustration of our approach. As can be seen, even though both n_1 and n_4 have similar *theme score*, final weights are different based on different *positional scores*.

In our experiments, the PageRank scores are updated until the difference between two consecutive iterations is ≤ 0.001 or for 100 iterations.

Candidate Phrases Formation

Candidate words appearing continuously in the document are concatenated to generate candidate phrases. We consider phrases with the regular expression (adjective)*(noun)+, of length up to four words, to generate candidate phrases. We used stemmed version of each word using Porter stemmer. We use POS tagger from Python's NLTK toolkit. The score for each candidate phrase is calculated by summing up the scores of its individual words[22]. The top-scoring phrases are output as predicted keyphrases for a given document.

2.3.3 Methods Used in CiteSeerX Case Study

Here we describe three supervised keyphrase extraction models that we explore to integrate into CiteSeerX: KEA[87], Hulth[21], and Citation-enhanced Keyphrase Extraction (CeKE)[11]. Unlike KEA and Hulth, which only use the title and abstract of a given research article, CeKE exploits citation contexts along with the title and abstract of the given document. A citation context is defined as the text within a window of n words surrounding a citation

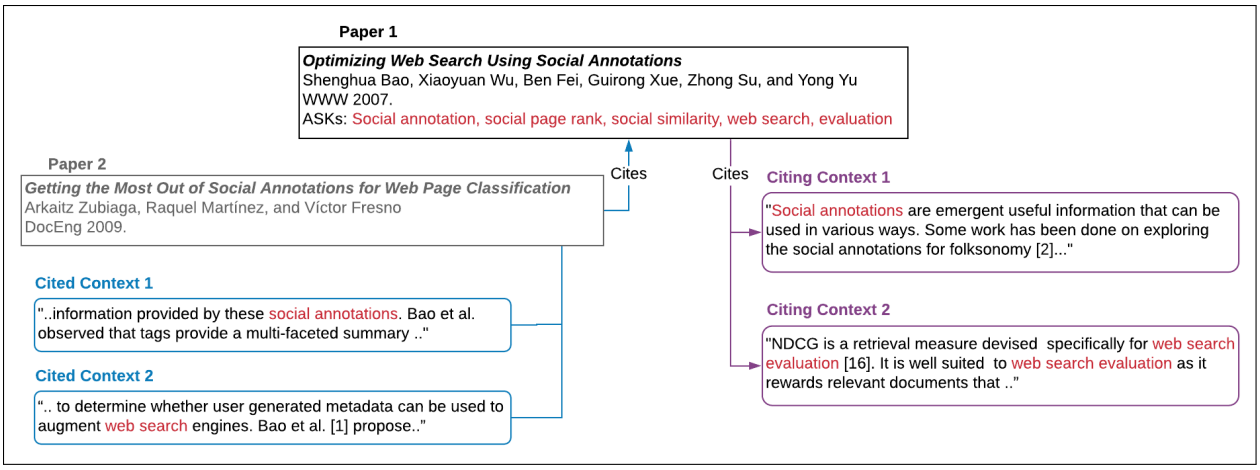


Figure 2.4: *A small citation network for Paper 1.*

mention. A citation context includes cited and citing contexts. A citing context for a target paper p is a context in which p is citing another paper. A cited context for a target paper p is a context in which p is cited by another paper. For a target paper, all cited contexts and citing contexts are aggregated into a single context.

Figure 2.4 shows an example of a small citation network using a paper (Paper 1) and its citation network neighbors. We can see the large overlap between the authors-submitted keyphrases and the citation contexts.

KEA: Frank et al.[87] used statistical features for the keyphrase extraction task and proposed a method named KEA. KEA uses following statistical features: *tf-idf*, i.e., the term frequency - inverse document frequency of a candidate phrase and the *relative position* of a candidate phrase, i.e., the position of the first occurrence of a phrase normalized by the number of words of the target paper. KEA extracts keyphrases from the title and abstract of a given paper.

Hulth: Hulth[21] argued that adding linguistic knowledge such as syntactic features can yield better results than relying only on statistics such as a term frequency (*tf*) and *n*-grams. Hulth showed remarkable improvement by adding part-of-speech (POS) tag as a feature along with statistical features. The features used in Hulth's approach are *tf*, *cf* (i.e., collection frequency), *relative position* and *POS tags* (if a phrase is composed by more than one word, then the POS will contain the tags of all words). Similar to KEA, Hulth extracts

keyphrases only from the title and abstracts.

Citation-Enhanced Keyphrase Extraction (CeKE): Caragea et al.[11] proposed CeKE and showed that the information from the citation network in conjunction with traditional frequency-based and syntactical features improves the performance of the keyphrase extraction models.

CeKE uses the following features: *tf-idf*; *relative position*; POS tags of all the words in a phrase; *first position* of a candidate phrase, i.e., the distance of the first occurrence of a phrase from the beginning of a paper; *tf-idf-Over*, i.e., a boolean feature, which is true if the *tf-idf* of a candidate phrase is greater than a threshold θ; *firstPosUnder*, also a boolean feature, which is true if the distance of the first occurrence of a phrase from the beginning of a target paper is below a certain threshold β. *Citation Network based features* include: *inCited* and *inCiting*, i.e., boolean features that are true if the candidate phrase occurs in cited and citing contexts, respectively; and *citation tf-idf*, i.e., the *tf-idf* score of each phrase computed from the aggregated citation contexts.

In our experiments, we compare three variants of CeKE: CeKE-Target that uses only the text from the target document; CeKE-Citing that uses the text from the target document and its citing contexts; CeKE-Cited that uses the text from the target document and its cited contexts; and CeKE-Both that uses both types of contexts.

2.4 Datasets

For evaluation of our proposed CRF-based approach and KPRank, we used five datasets of research papers for evaluation of our both proposed approaches. The first four datasets are widely used in keyphrase extraction and include SemEval-2010, Krapivin, Inspec, and NUS. The fifth dataset, called ACM, is a much larger dataset compared with the previous four datasets and is created from the collection of research papers published by ACM.[3] We used the title and abstract of each paper. Note that, for each dataset we use its test set for evaluation of unsupervised approaches. The five datasets are described below.

[3] https://dl.acm.org/

1. **SemEval**[88] contains 288 research papers from the ACM digital library along with author-assigned keyphrases. The dataset has a train and test split consisting of 188 and 100 papers, respectively.

2. **Krapivin**[89] contains 2,304 ACM research papers with full text and author-assigned keyphrases. Similar to[71], since the dataset does not have a train-test split, we sampled 400 papers as the test set with the remaining papers being used as the training set.

3. **Inspec**[21] contains abstracts of 2,000 research papers. It has a train-validation-test split of 1,000, 500 and 500 papers, respectively. In our experiments, we use the combination of train and validation sets to train different models.

4. **NUS**[46] contains 211 research papers. This dataset does not have a train and test split and it is relatively small. Hence, while testing supervised algorithms, consistent with[71], we performed five-fold cross-validation.

5. **ACM:** Since none of the above datasets is very large, we constructed a new dataset of 30,000 papers published in ACM conferences. These papers were sampled from the 211,028 papers available in ACM Digital Library and published after 2010. In our experiments, we used 10,000 papers (at random) as the train set (**ACM-10k**) and the remaining 20,000 papers as the test set (**ACM-20k**). During the dataset construction, we ensured that there was no overlap between our ACM dataset and any of the four datasets above.

Table 2.1 shows a summary of all five datasets and contains the number of papers in each dataset, the average number of keyphrases per paper, and the number of n-gram keyphrases, for $n = 1, 2, 3$, and $n > 3$, for each collection. The gold-standard for each paper contains the author-assigned keyphrases present in a paper (its title and abstract). For finding gold keyphrases in a paper, we used the stemmed version of each. Table 2.2 shows examples of gold keyphrases (shown for one paper) from each dataset. For the unsupervised approaches, we use the combination of controlled (author assigned) and uncontrolled (reader assigned) keyphrases as gold-standard phrases. We used uncontrolled keyphrases when available.

Dataset	Num. (#) Papers	Avg. # Keyphrases	Number of Keyphrases of Different Lengths
SemEval	288	7.15	#unigrams: 440, #bigrams: 769, #trigrams: 383, # > trigrams:145
Krapivin	2,304	3.03	#unigrams: 1,476, #bigrams: 3,598, #trigrams: 1,210, # > trigrams:328
Inspec	2,000	7.65	#unigrams: 2,153, #bigrams: 7,068, #trigrams: 4,050, # > trigrams:1,955
NUS	211	2.71	#unigrams: 183, #bigrams: 258, #trigrams: 76, # > trigrams:16
ACM-30k	30,000	2.67	#unigrams: 30,658, #bigrams: 36,017, #trigrams: 10,916, # > trigrams:2,489

Table 2.1: *Summary of the keyphrase extraction datasets used for model evaluation.*

Dataset	Keyphrases
SemEval	congestion game, load-dependent failure, identical resource, nash equilibrium, nondecreasing cost function, potential function, failure probability, load-dependent failure, pure strategy nash equilibrium
Kapivin	bessel function, modified Bessel function, order zero and one, vectorized software
Inspec	evolving fuzzy rule-based models, identification, noniterative update, rule-base structure, incremental unsupervised learning, ranking, informative potential, fuzzy rules, complex processes, air-conditioning component modeling
NUS	Nearest Neighbour Search, TLAESA, Approximation Search
ACM-30k	Control abstractions, Data abstractions, Programming languages, Programming methodology

Table 2.2: *Examples of gold standard keyphrases (present in the title + abstract) of a paper randomly selected from each dataset.*

For a case study on CiteSeerX, we created a dataset from CiteSeerX by matching $30,000$ randomly selected ACM papers against all CiteSeerX papers by title. We found $6,942$ matches. Among these papers, $6,942$, $5,743$, and $5,743$ papers have citing, cited, and both types of contexts, respectively. To create a dataset, we consider the documents for which we have both types of contexts and at least 3 author-supplied keyphrases appearing in titles or abstracts. We name this dataset as **ACM-CiteSeerX-KE**. Using these criteria, we identified 1,846 papers, which we used as our dataset for evaluation. The gold-standard contains the author-supplied keyphrases present in a paper (its title and abstract). Table 2.3 shows a summary of **ACM-CiteSeerX-KE** and contains the number of papers in the dataset, the average number of author-supplied keyphrases, and the number of n-gram author-supplied keyphrases, for $n = 1, 2, 3$, and $n > 3$.

ACM-CiteSeerX-KE					
Num. (#) Papers	Avg. # keyphrases	# keyphrases			
		#unigrams	#bigrams	#trigrams	# > trigrams
1,846	3.79	3,027	3,015	871	83

Table 2.3: *The dataset used during the case study on CiteSeerX.*

2.5 Experimental Design and Results

First, we show the experiments related to our proposed CRF-based supervised keyphrase extraction approach. Second, we show the experiments related to our proposed unsupervised keyphrase extraction approach KPRank. Lsatly, we show the case study on CiteSeerX.

For evaluation of our CRF-based supervised approach, we performed three types of experiments. First, we evaluate the quality of word embeddings by themselves or combined with document specific features and contrast the learned CRFs with the CRFs that do not use word embeddings. Second, we contrast the CRF model that uses word embeddings and document features with a more sophiticated and complex model, Bi-LSTM-CRF. In this experiment, we also study the performance of CRF and Bi-LSTM-CRF when trained on the ACM-10k dataset and evaluated on the test set of each of the four datasets, SemEval, Krapivin, Inspec, and NUS. Third, we compare the CRF against several baselines and prior works, including supervised, sequence labeling and neural models. Finally, we show anecdotal evidence that demonstrates the quality of word embeddings in extracting appropriate keyphrases.

For our sequence labeling task, for model training, we convert a pair of the paper (title and abstract) and keyphrases pairs such that each sentence of a paper is a sequence of word tokens, each token has a positive label (**KP**) if it occurs in a keyphrase in the gold-standard, or a negative label (**Non-KP**), otherwise. Predicted keyphrases are obtained by combining all consecutive words predicted as **KP** (i.e., the longest sequence). We did not impose a constraint on the length of keyphrases since, as we can see from Table 2.1, particularly for Inspec and ACM-30k datasets, the number of keyphrases with length greater than three is very large.

Evaluation metrics while comparison with CRF-based model. To evaluate the performance of the CRF models, we used the following metrics: precision, recall and F1-score for the positive class since the correct identification of positive examples (keyphrases) is more important. These metrics are widely used in previous works[11;21–23]. To match the predicted keyphrases with gold-standard keyphrases, we do exact match between the stemmed version of each. For the NUS we perform 5-fold cross validation and for the other four datasets we perform a single train and test (on the train-test split provided by the authors of each dataset, or our train-test split for ACM). Because NUS is very small, cross-validation experiments are more appropriate on this dataset[71]. For the 5-fold cross validation experiments, the reported values are averaged across all (document level) folds.

Evaluation metrics for KPRank and unsupervised approaches. To evaluate the performance of different unsupervised methods, we use micro avg. F1-score. We report the performance for the top 5 and 10 candidate phrases returned by different methods as in[71]. To create a word graph for a given document, we use its title and abstract. To match the predicted keyphrases with gold-standard keyphrases, we do exact match between the stemmed version of each.

2.5.1 Word Embeddings as Features in CRFs for Keyphrase Extraction

We contrast CRF models that use embedding features by themselves (EMB) and their combination with document specific features (EMB+DOC). We also contrast these models with the CRF models that use only document specific features (DOC) to understand the benefits of word embeddings in accurately predicting keyphrases. We also study the effect of embedding type (i.e., those trained on Google News and the ACM collection of research paper) and the embedding dimension.

We trained word embeddings of different dimension sizes (50, 100, 200, 300) on the ACM Digital Library collection of $211,028$ research papers using the Gensim implementation[90] of word2vec[39]. We kept words appearing in at least 5 documents for building the vocabulary.

	DOC				EMB				DOC+EMB			
	Pr%	Re%	F1%	Emb.	Dim.	Pr%	Re%	F1%	Dim.	Pr%	Re%	F1%
SemEval	**36.18**	61.26	45.49	ACM	300	27.02	43.38	33.30	300	33.74	65.14	44.46
				GNews	300	28.41	50.28	36.30	300	34.38	**68.29**	**45.73**
Krapivin	33.99	54.25	41.79	ACM	300	24.39	37.52	29.56	300	33.62	63.76	44.02
				GNews	300	29.69	26.49	28.00	300	**38.95**	**64.64**	**48.61**
Inspec	46.86	74.48	57.52	ACM	300	46.37	64.50	53.95	300	49.36	83.34	62.00
				GNews	300	49.64	59.53	54.14	300	**51.73**	**84.91**	**64.29**
NUS	**32.44**	**53.46**	**40.38**	ACM	50	19.09	30.91	23.60	50	25.50	43.75	32.22
				GNews	300	18.82	30.08	23.16	300	26.52	49.18	34.46
ACM	37.41	55.00	44.53	ACM	300	27.15	29.28	28.18	300	**39.50**	**63.56**	**48.72**
				GNews	300	14.50	8.65	10.83	300	38.40	54.99	45.22

Table 2.4: *CRF performance using word embeddings (EMB), document features (DOC) and DOC+EMB.*

The full text has around 1.2B tokens and 879K unique tokens. We compared the ACM word embeddings (denoted "ACM") with the pre-trained word2vec embeddings of 300-dimensional vectors on Google News of about 100B words (denoted "GNews").

Table 2.4 shows the results of these comparisons using ACM and GNews with the best performing embedding dimension, for all five datasets. For **SemEval**, **Krapivin**, **Inspec**, and **ACM**, we used the train-test split for model training and evaluation, as described in Section 2.4, whereas for **NUS**, we used five-fold cross-validation.

As can be seen from Table 2.4, word embeddings alone (EMB) perform worse than the document specific features alone (DOC), for all datasets, in terms of most compared measures. For example, on the ACM dataset, EMB achieves an F1-score of 28.19% (using ACM embeddings) as compared with 44.53% achieved by DOC. When we add word embeddings to document features (DOC+EMB), we can see substantial improvements in performance over the individual features, DOC or EMB alone, for Krapivin, Inspec, and ACM, in terms of all compared measures, regardless of the embedding type used, ACM or GNews. For example, on ACM, DOC+EMB achieves an F1-score of 48.72% (using ACM embeddings), whereas DOC and EMB alone achieve an F1-score of 44.53% and 28.18%, respectively. However, on SemEval and NUS, DOC alone performs similarly or better than DOC+EMB, e.g., on SemEval, DOC achieves an F1-score of 45.49% as compared to 45.73% achieved by DOC+EMB using GNews embeddings, whereas on NUS, DOC achieves an F1-score of 40.38% as compared to 34.46% achieved by DOC+EMB using GNews embeddings. One po-

ACM			Semeval			Krapivin			Inspec			NUS		
Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%
CRF														
39.5	**63.5**	**48.7**	**34.3**	**68.2**	**45.7**	**38.9**	**64.6**	**48.6**	**51.7**	**84.9**	**64.2**	**26.5**	**49.1**	**34.4**
Bi-LSTM-CRF														
36.0	50.0	41.9	23.1	28.3	25.4	22.8	42.3	29.6	27.6	19.6	23.0	24.5	47.8	32.6

Table 2.5: *Contrasting CRF with Bi-LSTM-CRF. The input to both models is DOC+EMB.*

tential explanation for this similar or drop in performance for SemEval and NUS is that both datasets have a relatively small size (see Table 2.1), with less than 200 papers for training in each dataset, that hinders to learn robust parameters for CRFs when the number of features is large as is the case with DOC+EMB, as compared with the number of DOC alone (only six features). These results show that word embeddings combined with document features help to improve the performance of CRF models when we have a sufficient amount of training data available.

From Table 2.4, we can also see that the embeddings trained on GNews perform similarly or better than those trained on ACM for DOC+EMB based CRFs on all datasets, except ACM. Also, for ACM embeddings, an embedding size of 300 generally works best among all compared sizes.

2.5.2 CRF vs. Bi-LSTM-CRF for Keyphrase Extaction

Next, we compare the CRF models with the more sophisticated Bi-LSTM-CRF models that are able to exploit the long-distance dependencies in the text. For the Bi-LSTM-CRF models, we used adam optimizer with learning rate 0.001, 300 cells for Bi-LSTM, and 30 epochs for model training.

Table 2.5 shows the results of this comparison for all five datasets. For model training and evaluation in this experiment, we used the train-test split of ACM, SemEval, Krapivin, and Inspec, and 5-fold cross-validation for NUS, with the embedding type and dimension that worked best on each dataset. The input of both CRFs and Bi-LSTM-CRFs is DOC+EMB. As can be seen from the table, CRFs consistently outperform the Bi-LSTM-CRFs on all five datasets. While the difference in F1-score is only 6.8% on ACM, it ranges between

	Semeval			Krapivin			Inspec			NUS		
Train: ACM-10k	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%
CRF	**43.9**	49.8	**46.7**	**37.8**	**58.0**	**45.7**	**42.5**	31.1	35.9	**38.2**	**67.3**	**48.7**
Bi-LSTM-CRF	32.2	**55.4**	40.7	27.8	54.1	36.7	36.0	**59.4**	**44.9**	26.4	56.4	35.9

Table 2.6: *Performance of DOC+EMB on SemEval, Krapivin, Inspec and NUS while training on ACM-10k.*

20% and 40% on SemEval, Krapivin, and Inspec, which is attributed to the small size of these three datasets. Moreover, the difference in F1-score on NUS is low compared with the other datasets (only 1.8%). We believe that both CRF and Bi-LSTM-CRF are incapable of learning good model parameters due to the very small size of NUS.

To understand the impact of the training set size on the performance of the CRF and Bi-LSTM-CRF models on SemEval, Krapivin, Inspec, and NUS, and determine which model is a better for keyphrase extraction, we performed the following experiment: we trained both CRFs and Bi-LSTM-CRFs on the ACM-10k training set and evaluated their performance on the test splits of each SemEval, Krapivin, Inspec, and on the NUS dataset. Table 2.6 shows the results of this experiment using ACM word embeddings (300 dimensional).

From Table 2.6, we can make several observations. First, the performance of Bi-LSTM-CRF increases in terms of all compared measures on all four datasets when we train the model on a much larger dataset size, i.e., 10,000 research papers, compared with the relatively small sizes of the training set of each dataset, supporting our intuition that the Bi-LSTM-CRF model overfits in the experiments in Table 2.5. A similar trend is observed on NUS for the CRF model, suggesting that the small size of NUS hinders both models to learn robust parameters. Second, the difference in F1-score between the CRF abd Bi-LSTM-CRF is smaller when we train the models on ACM-10k. Third, we observe that the F1-score of CRF is higher than that of Bi-LSTM-CRF on three datasets, SemEval, Krapivin, and NUS, but is lower on Inspec. Interestingly, on Inspec, Bi-LSTM-CRF achieves a higher recall as compared with CRF, i.e., 59.4% vs. 31.1%. Inspec has a much higher average number of keyphrases per paper (≈ 8 vs. 3) and contains a large number of keyphrases of length greater than 3 ($1,955$ n-grams with $n > 3$). Hence, the Bi-LSTM-CRF model, which exploits the long distance dependencies in the text, is able to accurately cover and identify these longer

keyphrases. A similar result on recall is observed on SemEval (see Tables 2.1 and 2.2 for datasets characteristics and gold keyphrases).

Moreover, on Inspec, the performance of CRF when trained on ACM-10k (Table 2.6) is consistently smaller than that of CRF when trained on Inspec itself (the training portion). We believe this is due to the distribution of author-assigned keyphrases, which are different for Inspec and ACM-10k, and the size of Inspec training set is good enough to train a good CRF model.

2.5.3 Baseline Comparisons for Our CRF-Based Supervised Keyphrase Extraction Model

Last, we compare the performance of CRF DOC+EMB using the best performing word embeddings and vector dimensions with several baseline approaches. Precisely, we compare the CRF DOC+EMB with six keyphrase extraction models: Hulth[21], KEA[24], Maui[48], the CRF model with posterior regularization (CRF/PR) from Gollapalli et al.[29], CopyRNN[71], and Key2Vec[32]. The choice of some of these models was motivated by their wide-spread usage as baselines in other related works[71], their good performance, and the integration of embeddings in existing systems as in Key2Vec. We used the implementations of Maui,[4] CRF/PR[29],[5] and CopyRNN,[6] and developed our implementation of Key2Vec.

Hulth uses POS tags, relative position, term frequency, and collection frequency as features. KEA uses tf-idf and relative position as features. Maui uses traditional and Wikipedia features, e.g., node degree, Wikipedia keyphraseness. The model proposed by Gollapalli et al.[29] uses CRF with posterior regularization, with three types of features: word, orthographic, and stopword features; parse-tree features; and title features, as well as their combinations incorporated with expert knowledge (i.e., predictions from other supervised and unsupervised methods). The CRF/PR model is the best performing model from Gollapalli et al.[29] and uses information solely from the document itself. CopyRNN uses an encoder-

[4] https://github.com/zelandiya/maui
[5] https://sites.google.com/site/sujathadas/home/pubslist
[6] https://github.com/memray/seq2seq-keyphrase

	SemEval			Krapivin			Inspec			NUS			ACM		
	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
CRF on DOC+EMB															
	34.4	**68.3**	**45.7**	**39.0**	**64.6**	**48.6**	51.7	**84.9**	**64.3**	26.5	49.2	34.5	**39.5**	**63.6**	**48.7**
Previous Models															
Hulth	28.7	10.7	15.6	21.2	8.4	12.1	27.5	70.5	39.6	20.2	12.4	15.4	20.0	14.2	16.6
KEA	25.6	10.0	14.4	22.2	10.1	13.9	26.6	5.1	8.5	25.3	24.2	24.7	22.6	17.0	19.4
Maui	27.4	38.1	31.9	20.0	56.0	29.5	24.2	34.2	28.4	20.3	54.7	29.6	22.0	63.2	32.7
CRF/PR	29.7	54.0	38.3	29.3	15.2	20.1	**56.9**	35.6	43.8	**33.2**	**61.4**	**43.1**	24.9	13.5	17.5
CopyRNN	24.7	39.5	30.4	17.6	54.2	26.6	29.2	40.9	34.1	26.6	42.1	32.6	17.8	42.2	25.0
Key2Vec	**38.6**	38.6	38.6	23.0	47.5	31.0	32.8	38.1	35.2	25.4	49.0	33.4	27.4	50.2	35.4

Table 2.7: *The comparison of CRF that uses DOC+EMB with previous works. Performance is shown in %.*

decoder Recurrent Neural Network with a copying mechanism as a generative model instead of extracting phrases only from the document text. We run CopyRNN for all five datasets with the parameter settings mentioned by the authors[71]. Key2Vec is a biased PageRank algorithm. We created a word graph by from nouns and adjectives and added an edge if two words appear within w words of each other in text. A *theme vector* for each paper was obtained by summing the embedding of the candidate words from its title. We used the GNews embeddings (300 dimension), which performed better than ACM in Key2Vec.

Table 2.7 shows the results of these comparisons for all five datasets, using the train-test split available for SemEval, Krapivin, Inspec, and ACM, and 5-fold cross-validation on NUS. As can be seen from the table, the DOC+EMB-based CRF that uses word embeddings in addition to document features substantially outperforms Hulth, KEA, Maui, CopyRNN, and Key2Vec, in terms of most compared measures, on four datasets, SemEval, Krapivin, Inspec, and ACM. For example, DOC+EMB based CRF achieves the highest F1-score of 64.3% on Inspec. Interestingly, on NUS, CRF/PR achieves the highest F1-score of 43.1% among all models. Moreover, CRF/PR achieves the highest precision on Inspec dataset, and the highest value among all measures on NUS. Key2Vec achieves the highest precision of 38.6% on SemEval. It is worth noting that the DOC+EMB based CRF models yield improvements in performance over more complex deep learning model, CopyRNN, on all five datasets.

Incorporating **site-level knowledge** to extract structured data from **web forums**

Web forums have become an important data resource for many web applications, but extracting **structured data** from unstructured **web forum** pages is still a challenging task [...]. In this paper, we study the problem of **structured data** extraction from various **web forum** sites. Our target is to find a solution as general as possible to extract **structured data**, such as post title, post author, post time, and post content from any forum site. In contrast to most existing **information extraction** methods, which only leverage the knowledge inside an individual page, we incorporate both page-level and **site-level knowledge** and employ **Markov logic networks** (MLNs) [...]. The experimental results on 20 forums show a very encouraging **information extraction** performance, and demonstrate the ability of the proposed approach on various forums. [...]

Incorporating site-level **knowledge** to extract structured data from **web forums**

Web forums have become an important **data** resource for many **web** applications, but **extracting structured data** from unstructured **web forum** pages is still a challenging task [...]. In this paper, we study the problem of structured **data extraction** from various **web forum sites**. Our target is to find a solution as general as possible to extract structured **data**, such as post title, post author, post time, and post content from any **forum site**. In contrast to most existing **information extraction** methods, which only leverage the knowledge inside an individual page, we incorporate both page-level and site-level **knowledge** and employ **Markov logic networks** (MLNs) [...]. The experimental results on 20 forums show a very encouraging **information extraction** performance, and demonstrate the ability of the proposed approach on various forums. [...]
Human-input keyphrases: *Web forums, Structured data, Information extraction, Site level knowledge, Markov logic networks*
Predicted keyphrases: ***Web forums, Information extraction, Markov logic networks,*** *Extracting structured data, Data, Knowledge, Data extraction, Web, Forum site, Web forum sites*

Figure 2.5: *The title, abstract, human-input keyphrases and predicted keyphrases of an ACM paper. The phrases marked with cyan in the title and abstract shown on the top of the figure are gold keyphrases, whereas the words and phrases marked with dark blue in the title and abstract shown on the bottom of the figure are predicted keywords/keyphrases.*

2.5.4 Anecdotal Evidence for Our CRF-Based Model

To see the quality of predicted phrases by the best EMB+DOC based CRF, we randomly selected a paper from our ACM dataset and evaluated the CRF model on it. Note that this selected paper belongs to the test portion of the dataset. We manually inspected the CRF predictions and contrasted them with the author-annotated (gold) keyphrases. The title, abstract, human annotated keyphrases and predicted keyphrases for this paper are shown in Figure 2.5. Specifically, the cyan bold phrases shown in the text on the top of the figure represent author-assigned keyphrases, whereas the dark blue bold phrases shown in the text

on the bottom of the figure represent the positively predicted phrases using the CRF trained on DOC+EMB features. It can be seen from the figure that the predicted keyphrases cover three out of five author assigned keyphrases, and for the remaining two author assigned keyphrases the CRF model predicted one super-string and one substring.

We can also observe that the model was not able to predict "structured data" nor "site level knowledge" even though both these gold keyphrases appear in the title of the document. However, for the two gold keyphrases missed by the model, the model predicted "extracting structured data", "data", and "knowledge" as keyphrases which are a super-string or substring of them. Moreover, predicted keyphrases "web", "forum site", and "web forum sites" are related to one of the author-assigned keyphrase "web forums," which was correctly predicted by the DOC+EMB CRF model. As there exist a semantic relation between "data" and "information," DOC+EMB is able to predict "data extraction," which is similar with the author-assigned keyphrase "information extraction".

2.5.5 KPRank: The Effect of Position, Contextual Embeddings, and the Comparison With Previous Works

To see the effect of positional information, we compare the performance of KPRank that uses contextual SciBERT (SB) embeddings along with positional information (denoted as KPRank(SB)) with that of its counterpart that does not use positional information (denoted as KPRank(SB−POS)). Moreover, to see the effect of contextual embeddings, we compare the performance of SciBERT-based KPRank (KPRank(SB)) with that of KPRank that uses FastText non-contextual word embeddings[43] (denoted as KPRank(FastText)). For FastText, we used pre-trained 300 dimensional embeddings trained on subword information on Common Crawl. Note that KPRank(SciBERT) and KPRank(FastText) use positional information along with the theme score. Last, we compare the performance of KPRank with Tf-Idf and six PageRank based unsupervised methods as baselines: PositionRank[33], Key2Vec[32], TextRank[23], SingleRank[22], ExpandRank[22], TopicRank[91].

Table 2.8 shows these comparisons on SemEval, Inspec, Krapivin, NUS, and ACM.

	SemEval		Inspec		Krapivin		NUS		ACM	
	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10
KPRank(SB)	**22.51**	**25.76**	**27.72**	32.30	17.74	18.57	**21.09**	**22.36**	**14.79**	**15.17**
KPRank(SB−POS)	17.33	23.97	26.88	32.72	15.81	16.97	16.79	19.68	12.13	13.14
KPRank(FastText)	22.04	25.39	27.28	32.12	18.20	**18.91**	20.69	22.12	14.77	15.08
PositionRank	**22.51**	24.99	26.73	31.84	**18.49**	18.30	18.65	20.99	13.04	14.09
Key2Vec	17.54	23.63	26.97	**32.82**	15.50	16.68	16.80	20.10	12.08	13.07
Tf-Idf	18.13	21.82	24.73	31.41	15.67	17.23	14.22	18.05	11.07	12.90
TextRank	12.12	17.90	22.81	30.47	09.15	12.91	09.04	12.64	05.02	07.54
SingleRank	11.22	18.24	24.11	31.96	10.25	13.53	08.69	13.78	05.66	08.32
ExpandRank	14.65	20.46	24.81	31.45	12.32	15.32	10.90	15.29	10.02	11.28
TopicRank	10.77	11.04	14.65	17.46	08.11	07.82	11.79	11.45	09.39	07.62

Table 2.8: *The comparison of SciBERT (SB) based KPRank, and previous works.*

It can be seen from the table that adding position information shows much higher improvement in the performance of KPRank, i.e. KPRank(SB) substantially outperforms KPRank(SB−POS). Moreover, KPRank(SB) outperforms KPRank(FastText) on all the datasets except for Krapivin. Importantly, KPRank(SB) outperforms most baseline methods, including Key2Vec (by a large margin) e.g., on SemEval, KPRank(SB) achieves an F1@5 of 22.51% as compared with 17.54% achieved by Key2Vec. We can also notice from Table 2.8 that KPRank(SB) achieves comparable performance whenever any baseline method achieves the best performance.

Figure 2.6 shows the confusion matrices of KPRank(SB) using @5 predictions on all five datasets. Each matrix is represented as a heat map, i.e., the darker the blue color the higher the value at that position and the darker the blue on the main diagonal, the more accurate the model is.

Comparison with a supervised approach. Usually, the performance of the supervised keyphrase extraction models is better than the unsupervised models[20]. We compare the performance of KPRank(SB) with the CRF-based sequence classification model for the keyphrase extraction[10] that uses word embeddings as features along with document specific features. The CRF model outperforms KPRank(SB) on all five datasets, e.g., CRF model achieves an F1 of 45.73% as compared with 25.76% achieved by KPRank(SB) on SemEval.

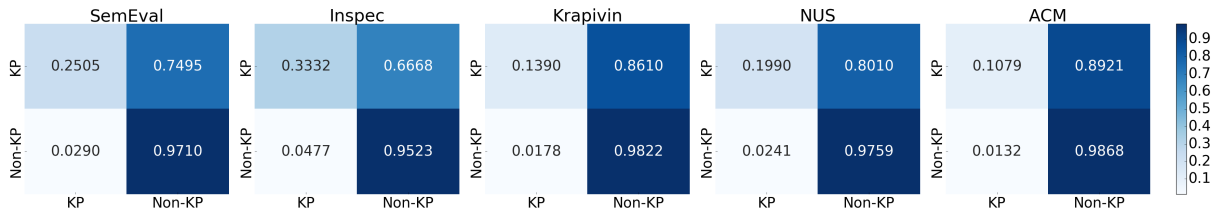


Figure 2.6: *Keyphrase extraction confusion matrices of KPRank(SB) using @5 predictions on all the datasets. The darker the blue on the main diagonal, the more accurate the model is.*

Organization design: The continuing influence of *information technology*

Drawing from an *information processing perspective*, this paper examines how *information technology* (IT) has been a catalyst in the development of new forms of *organizational structures*. [...] to the present *environmental instability* that now characterizes many industries. Specifically, the authors suggest that advances in IT have enabled managers to adapt existing forms and create new models for *organizational design* that better fit requirements of an unstable environment. [...]. IT has gone from a support mechanism to a substitute for *organizational structures* in the form of the shadow structure. [...]

Gold-standard keyphrases: *Organization design, Information processing perspective, Organizational structures, Environmental instability, Information technology*

Predicted keyphrases: ***Organization design***, ***Information technology***, ***Information processing perspective***, ***Organizational structures***, *Organizational design, Organization, Information processing, Shadow structure, New forms, Bureaucratic structure*

Figure 2.7: *The title, abstract, gold-standard keyphrases and predicted keyphrases of a paper. The phrases marked with cyan in the title and abstract shown on the top of the figure are gold-standard keyphrases.*

2.5.6 Anecdotal Evidence for KPRank

To see the quality of predicted phrases by the KPRank(SB), we randomly selected a paper from the Inspec dataset and evaluated the KPRank(SB) on it. We manually inspected the top-10 predictions by the KPRank(SB) and contrasted them with the gold-standard keyphrases. The title, abstract, gold-standard keyphrases and top-10 predicted keyphrases for this paper are shown in Figure 2.7. Precisely, in the figure, the cyan italic phrases shown in the text on the top of the figure represent gold-standard keyphrases, whereas the bottom of the figure shows gold-standard keyphrases and the top-10 predicted keyphrases by KPRank(SB) (shown in the order of their prediction). It can be seen from the figure that

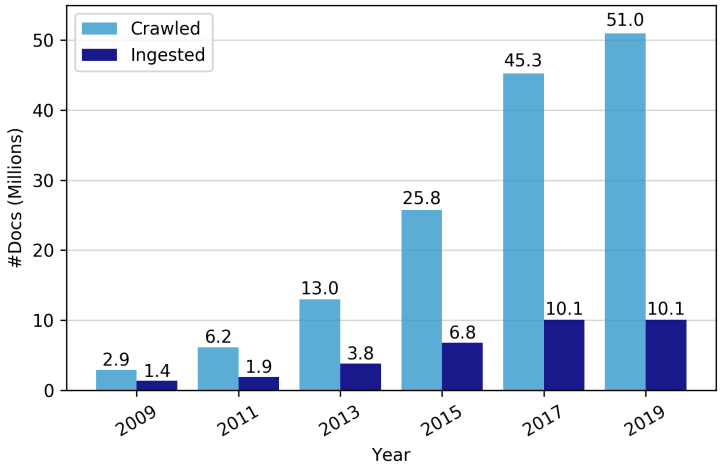


Figure 2.8: *Number of documents crawled and ingested from past few years in CiteSeerX.*

four out of five gold-standard keyphrases are present in the top-5 predicted keyphrases.

We can also see that KPRank(SB) did not predict gold-standard phrase "environmental instabily." A closer inspection of the document and both types of scores (*theme score* and *positional score*) assigned by KPRank(SB) to both constituent words of the gold-standard phrase that was not ranked in top-10 predictions revealed that these constituent words have lower values of *theme score* and they both appear only once in the document. Hence, the Pagerank algorithm will not boost these words. Inspecting other errors, we found that KPRank can fail to predict phrases that contain words that are less frequent in the document and their word embeddings are far from the *theme vector*.

2.6 Keyphrase Extraction in CiteSeerX

There are in general two types of digital library search engines. The first type obtains publications and metadata from publishers, such as ACM Digital Library, IEEE Xplore, and Elsevier. The other type, such as CiteSeerX[92], crawls the public Web for scholarly documents and *automatically* extracts metadata from these documents.

CiteSeer was launched in 1998[92] and its successor CiteSeerX[93] has been online since 2008. Since then, the document collection has been steadily growing (see Figure 2.8). The goal of

CiteSeerX is to improve the dissemination of and access to academic and scientific literature. Currently, CiteSeerX has 3 million unique users world-wide and is hit 3 million times a day. CiteSeerX reaches about 180 million downloads annually[94]. Besides search capabilities, CiteSeerX also provides an Open Archives Initiative (OAI) protocol for metadata harvesting. CiteSeerX receives about 5,000 requests per month to access the OAI service. Researchers are interested in more than just CiteSeerX metadata. For example, CiteSeerX receives about 10 requests for data per month via the contact form on the front page[95]. These requests include graduate students seeking project datasets and researchers that were looking for large datasets for experiments. CiteSeerX hosts a dump of the database and other data on Google Drive.

In the early stage, the crawl seeds were mostly homepages of scholars in computer and information sciences and engineering (CISE). In the past decade, CiteSeerX added to the crawls seed URLs from the Microsoft Academic Graph[96], and directly incorporated PDFs from PubMed, arXiv, and digital repositories in a diverse spectrum of disciplines. A recent work on subject category classification of scientific papers estimated that the fractions of papers in physics, chemistry, biology, materials science, and computer science are 11.4%, 12.4%, 18.6%, 5.4%, and 7.6%, respectively[97]. CiteSeerX is increasing its document collection by actively crawling the Web using new policies and seeds to incorporate new domains. We expect this to encourage users from multiple disciplines to search and download academic papers and to be useful for studying cross discipline citation and social networks.

Since CiteSeerX was developed, many artificial intelligence techniques have been developed and deployed in CiteSeerX[93], including but not limited to header extraction[98], citation extraction[99], document type classification[100], author name disambiguation[101], and data cleansing[102]. In addition, an unsupervised NP-Chunking method is deployed for automatic keyphrase extraction. Besides author-submitted keyphrases, CiteSeerX extracts on average 16 keyphrases per paper using NP-Chunking. Users can search for a particular keyphrase by clicking it. This feature provides a shortcut for users to explore scholarly papers in related topics of the current paper they are browsing. All automatically extracted keyphrases are displayed on the summary page, and they deliver detailed domain knowledge in scholarly

Year	#Docs. (Millions)	#Keyphrase-Clicks (Millions)	#Unique-Keyphrases (Millions)
2016	8.44	4.41	1.60
2017	10.1	7.17	1.86
2018	10.1	7.52	1.74

Table 2.9: *The number of full text documents, the total number of keyphrase-clicks, and unique keyphrases clicked for years 2016, 2017, and 2018 in CiteSeerX.*

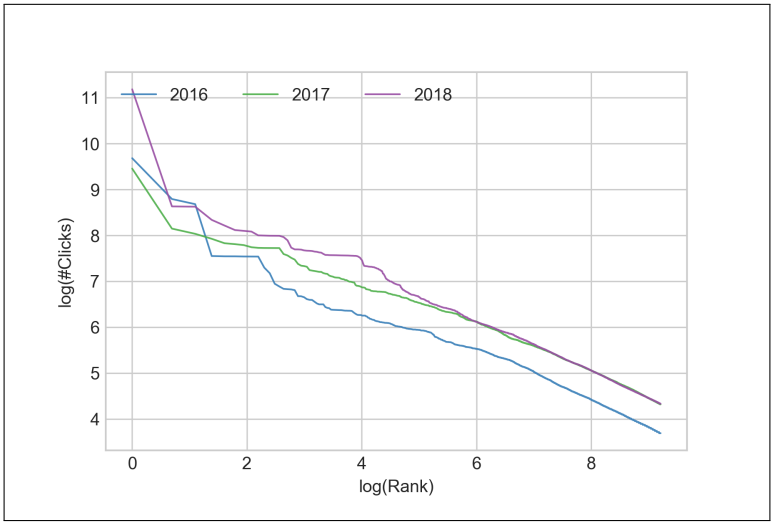


Figure 2.9: *log(Rank) vs log(Clicks) for top-*$10,000$ *keyphrases clicked by users of CiteSeerX during years 2016, 2017, and 2018.*

documents. Every time a keyphrase is clicked, CiteSeerX searches the clicked keyphrase and refreshes the search results. To understand how keyphrases promote paper browsing and downloading, we analyze the access logs retrieved from three web servers from 2016 to 2018.

2.6.1 Click-log Analysis

Table 2.9 shows the total number of documents, keyphrase clicks, and unique keyphrases clicked from 2016 to 2018. The total number of keyphrase clicks increased significantly by $\sim 63\%$ from 2016 to 2017. For years 2017 and 2018, although the total number of documents stayed about the same (10.1 million), the total number of keyphrase clicks increased by 5%. Although there is a slight decrease in the number of unique keyphrases clicked, the increase in the number of keyphrase clicks from year 2016 to year 2018 showcases the increasing use and the popularity of keyphrases.

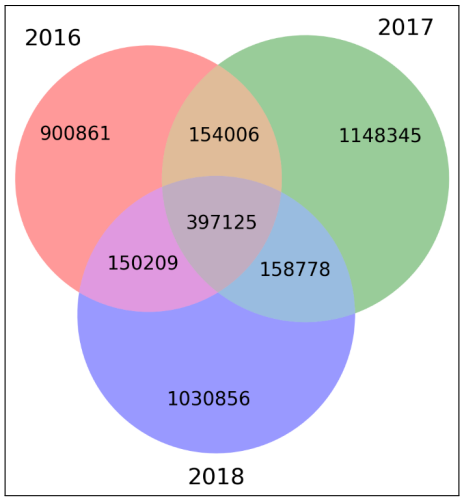


Figure 2.10: *Venn Diagram for all 3 years based on unique keyphrases.*

Figure 2.9 shows the ranking versus the number of clicks (#clicks) in logarithmic scale for the 10,000 most popular keyphrases during the three years. We can see that the #click decreases exponentially as the rank increases, which mimics the Zipf's law for all three years.

Figure 2.10 shows the Venn diagram for the unique keyphrases clicked during years 2016, 2017, and 2018. As seen from the figure, in two consecutive years, only about one third of the keyphrases are common, whereas two third of the keyphrases are new. For example, 1.6 million unique keyphrases were clicked in 2016 but only about $551k$ (33%) were carried to 2017. Similarly, 1.86 million unique keyphrases were clicked in 2017, but only $555k$ (30%) were carried out in 2018. This trend implies that user interests have been rapidly evolving over these years, but there is still a considerable number of topics searched among several years. These conclusions are made based on the analysis of open-access documents from a three years time period. However, further analysis is needed for more comprehensive conclusions.

Table 2.10 shows the top-20 most frequent keyphrases clicked. We can see that the extracted keyphrases are not always terminological concepts as seen usually in author-submitted keyphrases. Examples such as "local", "experimental results", "wide range", and "recent year" were extracted just because they are noun phrases. This indicates that more sophisticated models are necessary to improve the quality of extracted keyphrases. It is interesting that these phrases were highly clicked, but investigating the reason is beyond

Year	Keywords
2016	DgNe, local, bullying, violence, bullied, bully, aggressive, aggression, R. Nobrega, experimental result, data, wide range, machine, lpEu, dvd, last year, recent year, artificial intelligence, key word, new technology
2017	key word, experimental result, wide range, large number, string theory, bullying, violence, bullied, bully, aggressive, aggression, recent year, new method, artificial intelligence, important role, machine learning, neural network, online version, environmental protection agency, wide variety
2018	JMQi, experimental result, key word, large number, wide range, aggression, violence, bullying, bully, bullied, aggressive, recent year, case study, wide variety, different type, sustainable development, informational security, VWBc, sensor network, simulation result

Table 2.10: *Top-20 keyphrases clicked during years 2016, 2017, and 2018.*

the scope of this paper.

2.6.2 Experiments and Results

During CiteSeerX case study, we generate candidate phrases for each document by applying POS filters. Consistent with previous works[11;21–23;60], these candidate phrases are identified using POS-tags of words, consisting of only nouns and adjectives. We apply Porter stemmer on each word. The initial position of each word is kept before removing any words. Second, to generate candidate phrases, contiguous words extracted in the first step are merged into n-grams ($n = 1, 2, 3$). Finally, we eliminate candidate phrases that end with an adjective and unigrams that are adjectives[11;22].

Evaluation metrics. To evaluate the performance of the keyphrase extraction methods, we use the following metrics: precision, recall and F1-score for the positive class since the correct identification of positive examples (keyphrases) is more important. The reported values are averaged in 10-fold cross-validation experiments, where folds were created at document level and candidate phrases were extracted from the documents in each fold to form the training and test sets. In all experiments, we used Naïve Bayes on the feature vectors extracted by each model.

Table 2.11 shows the performance of NP-Chunking, KEA, Hulth, CeKE-Target, CeKE-Citing, CeKE-Cited, and CeKE-Both. The table shows the evaluation measures and time

Model	Pr (%)	Re (%)	F1 (%)	Time/Doc (Sec)
NP-Chunking	04.26	29.19	07.44	**1.01**
Hulth	25.91	16.15	19.86	4.47
KEA	**30.41**	20.78	24.65	4.53
CeKE-Target	27.31	35.57	30.86	4.69
CeKE-Citing	25.65	40.45	31.37	6.61
CeKE-Cited	26.49	**42.73**	**32.68**	7.14
CeKE-Both	25.07	42.19	31.42	7.97

Table 2.11: *The comparison of different models using 10-fold cross-validation on* ***ACM-CiteSeerX-KE****.*

taken by each method using 10-fold cross-validation on **ACM-CiteSeerX-KE**. In NP-Chunking, the given text is first tokenized and tagged by a POS tagger. Based on the POS-tagging result, a grammar-based chunk parser is applied to separate two types of phrase chunks: (1) nouns or adjectives, followed by nouns (e.g., "relational database" or "support vector machine"), and (2) two chunks of (1) connected with a preposition or conjunction (e.g., "strong law of large numbers"). Time is measured on a computer with Xenon E5-2630 v4 processor and 32GB RAM. In CiteSeerX, the header extraction tool can extract the title, abstract, and citing contexts for a target document. However, to extract cited contexts in CiteSeerX, there is an overhead of 1.2 seconds per document on average to search and extract it from the CiteSeerX database.

It can be seen from Table 2.11 that, CeKE-Cited achieves the highest recall and F1 of 42.73% and 32.68%, respectively. KEA achieves the highest precision of 30.41% compared with other models with top-5 predictions. NP-Chunking takes the shortest time of 1.01 seconds to extract keyphrases from a document. However, NP-Chunking suffers from low precision and F1. CeKE variants outperform Hulth and KEA in terms of recall and F1, i.e., CeKE-Citing achieves an F1 of 32.68% as compared with 24.65% achieved by KEA. Moreover, CeKE variants that make use of citation contexts outperform CeKE-Target that does not use any citation contexts.

It can be seen from the table that CeKE-Cited achieves highest F1 of 32.68%. However, CeKE-Citing takes less time compared with CeKE-Cited, i.e., CeKE-Citing takes 6.61 seconds on average per document compared with 7.14 seconds taken by CeKE-Cited. CeKE-

Title: Incorporating **site-level knowledge** to extract structured data from **web forums**

Abstract: **Web forums** have become an important data resource for many web applications, but extracting **structured data** from unstructured **web forum** pages is still a challenging task [...]. In this paper, we study the problem of **structured data** extraction from various **web forum** sites. Our target is to find a solution as general as possible to extract **structured data**, such as post title, post author, post time, and post content from any forum site. In contrast to most existing **information extraction** methods, which only leverage the knowledge inside an individual page, we incorporate both page-level and **site-level knowledge** and employ **Markov logic networks** (MLNs) [...]. The experimental results on 20 forums show a very encouraging **information extraction** performance, and demonstrate the ability of the proposed approach on various forums. [...]

Author-supplied keyphrases: *Web forums, Structured data, Information extraction, Site level knowledge, Markov logic networks*

CeKE-Citing predicted keyphrases: ***web forum***, ***Site Level Knowledge***, *forum,* ***structured data***
Hulth predicted keyphrases: *forum, page, Knowledge, post,* ***Site Level Knowledge***, ***web forum***, ***structured data***
KEA predicted keyphrases: ***Site Level Knowledge***, ***web forum***, *forum, post*

Figure 2.11: *The title, abstract, author-supplied keyphrases and predicted keyphrases of an ACM paper. The phrases marked with cyan in the title and abstract shown in the figure are author-supplied keyphrases.*

Citing and CeKE-Both achieve comparable F1 of 31.37% and 31.42%, respectively. In terms of speed, CeKE-Target is the fastest among other variants because it does not need to perform POS tagging for citation contexts. Citing contexts can be extracted relatively straightforward from the content of the document. On the other hand, to extract cited contexts, we need the citation graph, from which we can obtain documents citing the target paper. We plan to select CeKE-Citing to deploy along with Hulth and KEA for the following reasons: CeKE-citing is faster than CeKE-cited and CeKE-Both; extracting cited contexts has an extra overhead to find it within a citation network; and cited context may not be present for all the articles.

Anecdotal Example:

To demonstrate the quality of extracted phrases by different methods (CeKE-Citing, Hulth, and KEA), we select an ACM paper at random from the testing corpus and manually com-

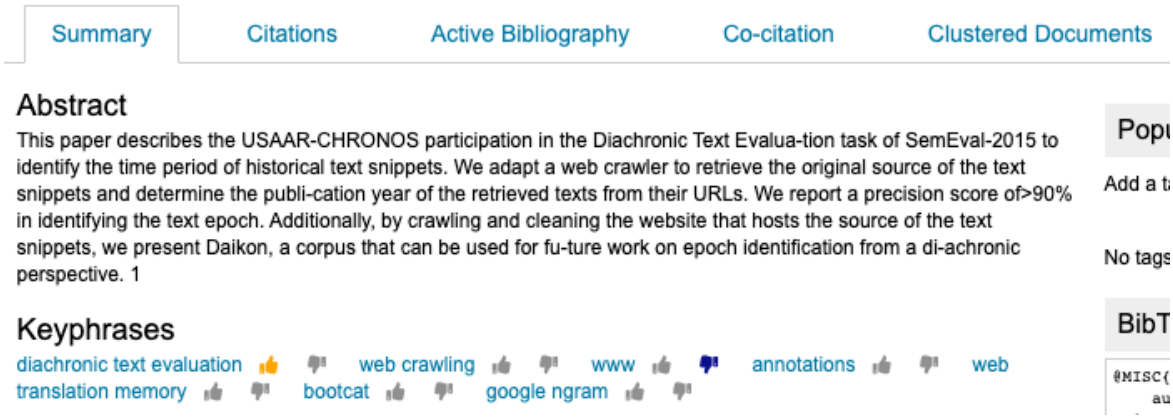


Figure 2.12: *A clip of a portion of a CiteSeerX paper's summary page containing a "Keyphrase" section that displays keyphrases extracted. Each keyphrase has a thumbup and a thumbdown button. A logged in user can vote by clicking these buttons.*

pared the keyphrases extracted by the three methods and the author-supplied keyphrases (Figure 2.11). Specifically, the cyan bold phrases shown in the text on the top of the figure represent author-supplied keyphrases, whereas the bottom of the figure shows author-supplied keyphrases and predicted keyphrases by each evaluated model. It can be seen from the figure that the CeKE-Citing predicted four keyphrases out of which three are ASKs. Hulth predicted seven keyphrases out of which three are author-supplied keyphrases. KEA predicted three keyphrases out of which two belong to author-supplied keyphrases. The predicted keyphrases by all three models that do not belong to author-supplied keyphrases are single words. This example demonstrates that CeKE-citing exhibits a better performance than the other two models.

2.6.3 Crowd-sourcing

The comparison between different keyphrase extraction models relies on ground truth datasets compiled from a small number of papers. We propose to evaluate keyphrase extraction models using crowd-sourcing, in which we allow users to vote for high quality keyphrases on papers' summary pages in CiteSeerX. These keyphrases are extracted using different models, but the model information is suppressed to reduce judgment bias. Voting systems are ubiquitous in social networks and multimedia websites, such as Facebook and YouTube, but

they are rarely seen in scholarly digital libraries. A screenshot of an example of the voting interface is shown in Figure 2.12. A database is already setup to store the total number of counts for each voting type as well as each voting action. The database contains the following tables.

- **Model table.** This table contains information of keyphrase extraction models.
- **Voting table.** This table contains the counts of upvotes and downvotes of keyphrases extracted using all models from all papers. The table also records the time the voting of a keyphrase is last updated. The same keyphrase extracted by two distinct models will have two entries in this table.
- **Action table.** This table contains information of all voting actions on keyphrases, such as the action time, the type of action (upvote vs. downvote), the IDs of keyphrases voted, and the IDs of voters. A voter must log in first before they can vote. If a voter votes a keyphrase extracted by two models, two actions will be recorded in this table. If a user reverses his vote, two actions (unvote and vote) are recorded in this table.

The extraction modules can be evaluated by the summation of eligible votes over all papers. In classic supervised machine learning, predicted keyphrases are evaluated by comparing extraction results against the author-supplied keyphrases[11]. However, the list of author-supplied keyphrases may not be exhaustive, i.e., certain pertinent keyphrases may be omitted by authors, but extracted by trained models. Crowd-sourcing provides an alternative approach that evaluates the pertinence of keyphrases from the readers' perspectives. However, there are certain potential biases that should be considered when deploying the system. One factor that can introduce bias is ordering because voters may not go through the whole list and vote all items. To mitigate this bias, we will shuffle keyphrases when displaying them on papers' summary pages. Another bias is the "Mathew's Effect" in which items with higher votes tend to receive more upvotes. We will hide the current votes of keyphrases to mitigate this effect.

We plan to collect votes after opening the voting system for at least 6 months. Using this

approach, the keyphrase extraction models can be evaluated at two levels. At the *keyphrase level*, we only consider keyphrases with at least 10 votes and apply a binary judgment for keyphrase quality. A keyphrase is "favored" if the number of upvotes is higher than the downvotes, otherwise, it is labeled as "disfavored". We can then score each model based on the number of favored vs. disfavored. At the *vote level*, we can score each model using upvotes and downvotes of all keyphrases. The final scores should be normalized by the number of keyphrase extracted by a certain model and voted by users.

2.6.4 Development and Deployment

Although CiteSeerX utilizes open source software packages, many core components are not directly available from open source repositories and require extensive programming and testing. The current CiteSeerX codebase inherited little from its predecessor's (CiteSeer) for stability and consistency. The core part of the main web apps were written by Dr. Isaac Councill and Juan Pablo Fernández-Ramírez and many components were developed by other graduate students, postdocs and software engineers, which took at least 3-4 years.

CiteSeerX has been using keyphrases extracted using an unsupervised NP-Chunking method. This method is fast and achieves high recall, but it has a relatively low precision. Thus, we are exploring supervised models to extract keyphrases more accurately into CiteSeerX. Our keyphrase extraction module employs three methods: CeKE, Hulth, and KEA. The keyphrase extraction module runs on top of several dependencies, which handle metadata extraction from PDF files and document type classification in CiteSeerX. For example, GROBID[103] is used to extract titles, abstracts, and citing contexts. We also developed a program to extract cited contexts for a given article from the CiteSeerX database. In addition, a POS tagger[7] is a part of our keyphrase extraction module and is integrated in the keyphrase extraction module. Even though we selected CeKE-Citing, the keyphrase extraction package supports other variants of CeKE and it is straightforward to switch between them. Figure 2.13 and Figure 2.14 show the CiteSeerX system architecture and schematic

[7]We have used NLP Stanford part of speech tagger.

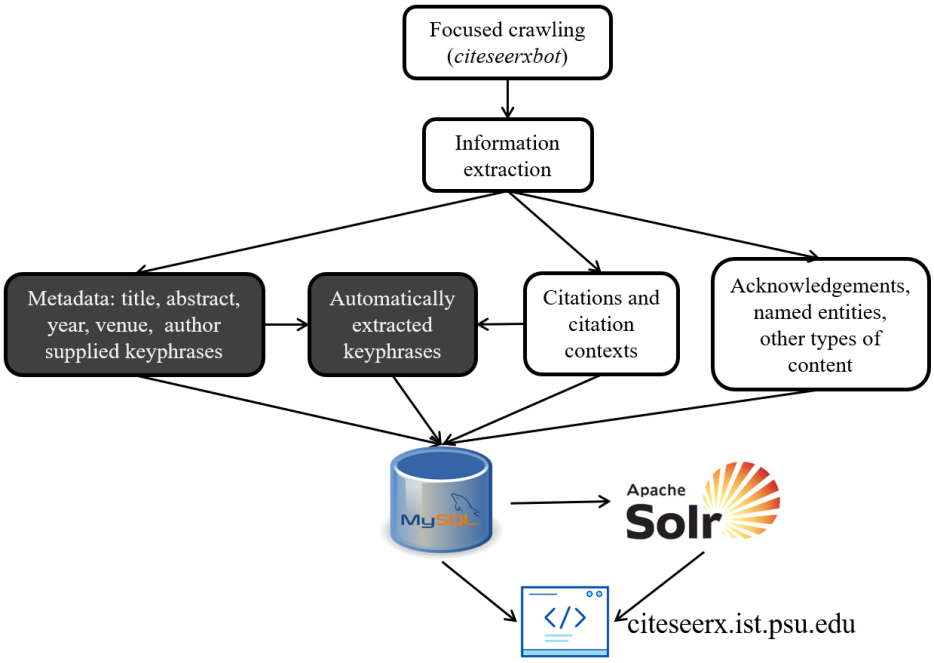


Figure 2.13: *CiteSeerX architecture.*

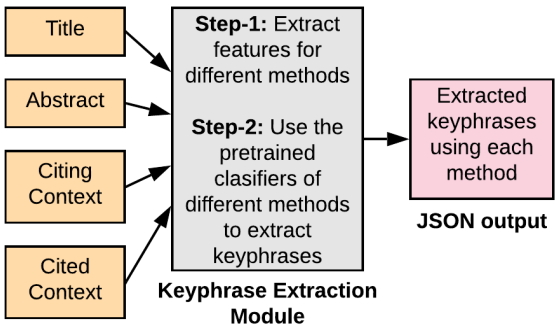


Figure 2.14: *Schematic diagram of keyphrase extraction module.*

diagram of our keyphrase extraction module, respectively.

2.6.5 Maintenance

The keyphrase extraction module is developed and maintained by about 3 graduate students and a postdoctoral scholar in an academic setting. The keyphrase extraction project received partial financial support from the National Science Foundation. The maintenance work includes, but is not limited to fixing bugs, answering questions from GitHub users, updating extractors with improved algorithms, and rerunning new extractors on existing

papers. Specific to the keyphrase extraction module, it can easily integrate new models trained on different or large data for the existing methods. In future, we aim to integrate new keyphrase extraction models. The key bottleneck is to integrate keyphrase modules into the ingestion system, so both author-supplied keyphrases and predicted keyphrases can be extracted with other types of content at scale. One solution is to encapsulate keyphrase extraction modules into Java package files (.jar files) or Python libraries so they can easily be invoked by PDFMEF[104], a customizable multi-processing metadata extraction framework for scientific documents. Currently, the CiteSeerX group is developing a new version of digital library framework that employs PDFMEF as part of the information extraction pipeline. The encapsulation solution can potentially reduce the maintenance cost and increase modularity.

2.7 Summary and Future Directions

In this chapter, we explored keyphrase extraction as sequence labeling using CRFs and studied the benefits of using word embeddings in conjunction with document specific features in order to capture the semantics of words in context. Our results showed interesting performance variability, according to training set sizes, the number of keyphrases per document, and their length in the number of words, but clearly highlighted the benefits of using word embeddings as features in CRFs along with document specific features. Our results also showed improvements in performance over complex models including more sophisticated deep learning models. Moreover, we proposed a novel unsupervised graph-based algorithm, KPRank, that incorporates both positional appearances of the words along with contextual word embeddings for computing a biased PageRank score for each candidate word. Our experimental results on five datasets show that incorporating position information into our biased KPRank model yields better performance compared with a KPRank that does not use the position information, and SciBERT-based KPRank usually outperforms FastText-based KPRank on this task. Moreover, KPRank outperforms strong baseline methods. By analyzing access logs of CiteSeerX in the past 3 years, we found that there are 3% of keyphrases common across all years, while there are many keyphrases which are only clicked during

a particular year. In the CiteSeerX case study, we proposed to integrate three supervised keyphrase extraction models into CiteSeerX which are more robust than the previously used NP-Chunking method. To evaluate the keyphrase extraction methods from a user perspective, we implemented a voting system on papers' summary pages in CiteSeerX to vote on predicted phrases without showing the model information to reduce potential judgment bias from voters.

In the future, it would be interesting to integrate posterior regularization in the word embeddings based CRF models. It would also be interesting to explore keyphrase extraction from research papers from other fields in Computer Science such as Computational Linguistics, as well as other scientific domains, such as Biology, Social Science, Political Science, and Material Sciences. Moreover, since these scientific domains do not generally have author-annotated keyphrases, developments of domain adaptation and transfer learning techniques should also be investigated. It would be interesting to explore KPRank on other domains, such as Biology, and Social Science. Moreover, it would be fascinating to explore integration of other keyphrase extraction models as well as other information extraction tools such as name-entity extraction tool into CiteSeerX to improve the user experience.

Chapter 3

Applications of Keyphrases/Keywords

Scholarly digital libraries provide access to scientific publications and comprise useful resources for researchers. Despite the advancements in search engine features, ranking methods, technologies, and the availability of programmable APIs, current-day open-access digital libraries still rely on crawl-based approaches for acquiring their underlying document collections. In this chapter, we propose a novel search-driven framework for acquiring documents for such scientific portals. Keyphrases or keywords are very useful to formulate queries that can retrieve topically-related articles from the Web. Within our framework, publicly-available research paper titles (keywords) and author names are used as queries to a Web search engine. We were able to obtain $\approx 267,000$ unique research papers through our fully-automated framework using $\approx 76,000$ queries, resulting in almost $200,000$ more papers than the number of queries. Moreover, through a combination of title and author name search, we were able to recover 78% of the original searched titles.

Furthermore, We propose a novel search-driven approach to build and maintain a large collection of homepages that can be used as seed URLs in any digital library including CiteSeerX to crawl scientific documents. Precisely, we integrate Web search and classification in a unified approach to discover new homepages: first, we use publicly-available author names and research paper titles (keywords) as queries to a Web search engine to find relevant content, and then we identify the correct homepages from the search results using a

powerful deep learning classifier based on Convolutional Neural Networks. Moreover, we use *Self-Training* in order to reduce the labeling effort and to utilize the unlabeled data to train the efficient researcher homepage classifier. Our experiments on a large scale dataset highlight the effectiveness of our approach, and position Web search as an effective method for acquiring authors' homepages. We show the development and deployment of the proposed approach in CiteSeerX and the maintenance requirements.

3.1 Introduction

Scientific portals such as Google Scholar, Semantic Scholar, ACL Anthology, CiteSeer$^{\mathtt{x}}$, and ArnetMiner, provide access to scholarly publications and comprise indispensable resources for researchers who search for literature on specific subject topics. Moreover, many applications such as document and citation recommendation[1–3], expert search[4;5], topic classification[6;7], and keyphrase extraction and generation[8–11], involve Web-scale analysis of up-to-date research collections.

Open-access, autonomous systems such as CiteSeer$^{\mathtt{x}}$ and ArnetMiner acquire and index freely-available research articles from the Web[105;106]. In order to enlarge its document collection, CiteSeerX maintains whitelists / blacklists of URLs and lists of researchers' homepages to direct the crawl for documents. Thus, maintaining comprehensive, up-to-date collections of researchers' homepages is an essential component in CiteSeerX. However, this task is very challenging since not only do new authors emerge, but also existing authors may stop publishing or may change affiliations, resulting in outdated ($4XX$ error) or invalid URLs. An analysis of a subset of $13,239$ homepages available in DBLP revealed that $\approx 42\%$ of them were outdated within a time span of three years. This represents about half of the original homepages in the set. *Given this challenge, how can we automatically augment the document collections and accurate lists of researchers' homepages in open-access scientific portals?*

For automatically augmenting the document collections, we propose a novel framework based on Web search. To motivate our framework, we recall how a Web user typically searches for research papers or authors. As with regular document search, a user typically

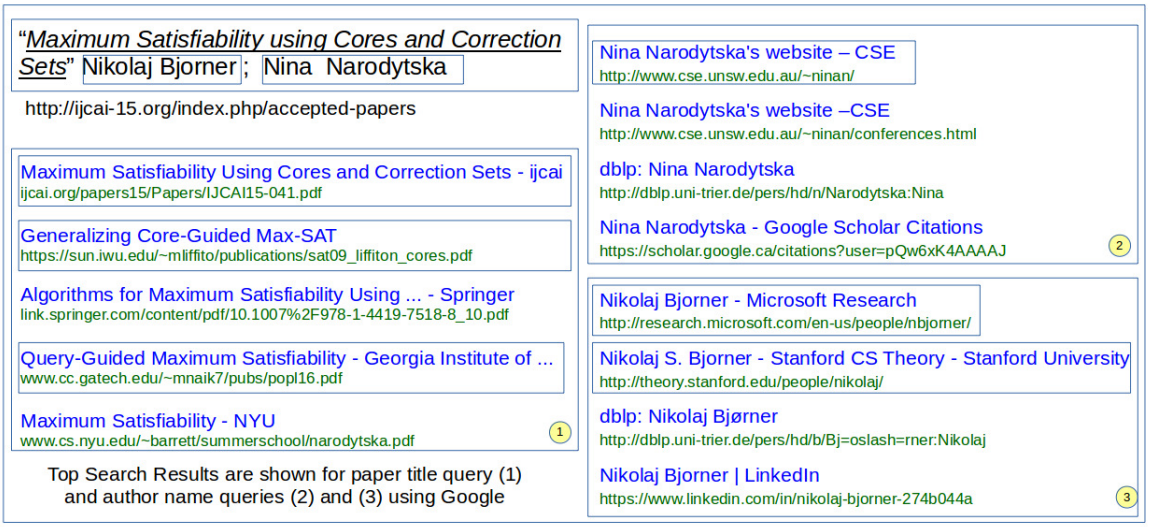


Figure 3.1: *An anecdotal search example for illustration.*

issues Web search queries comprising of representative keywords or paper titles for finding publications on a topic. Similarly, if the author is known, a "navigational query" [27] may be employed to locate the homepage where the paper is likely to be hosted. To illustrate this process, Figure 3.1 shows an anecdotal example of a search using Google for the title and authors of a research article. As can be seen from the figure, the intended research paper and the researchers' homepages (highlighted in sets 2 and 3) are accurately retrieved. Moreover, among the top-5 results shown for the title query (set 1), four of the five results are research papers on the same topic (i.e., the first four results). The document at the Springer link is not available for free, whereas the last document corresponds to course slides. The additional three papers are potentially retrieved because scientific paper titles comprise a large fraction of keywords [107], and hence, the words in these titles serve as excellent keywords that can retrieve not only the intended paper, but also other relevant documents. Our research papers discovery framework mimics precisely the above search and scrutinize the approach adopted by Scholarly Web users. Freely-available information from the Web for specific subject disciplines[1] is used to frame title and author name queries in our framework.

For maintaining the accurate list of researchers' homepages, one approach would be to crawl academic websites (e.g., from a university domain) and use a machine learning

[1]For example, from bibliographic listings such as DBLP or paper metadata available in ACM DL.

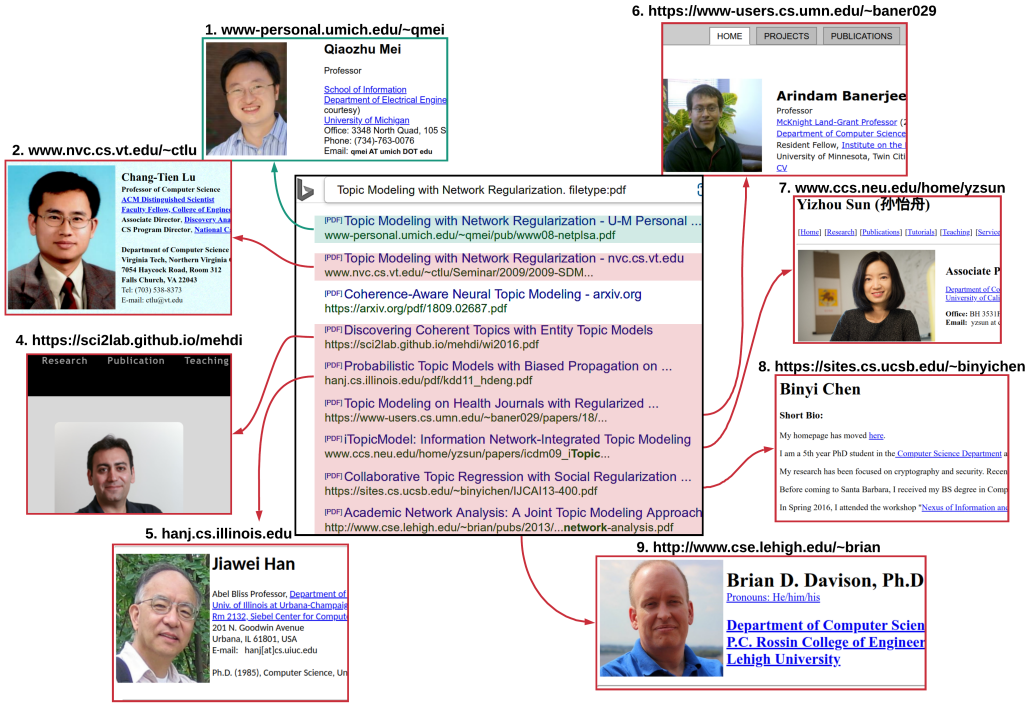


Figure 3.2: *An anecdotal search example using paper title search. Green highlighted response is located on the first author's homepage. Newly discovered authors are highlighted by a red color.*

classifier to predict whether a website accessed during the crawl is an author homepage or not[28]. However, this approach: (1) is still inefficient (e.g., in terms of bandwidth and storage resources) since only a small fraction of the websites hosted in an academic domain are author homepages (with many websites corresponding to departments, courses, groups, etc), and (2) misses homepages from research industry labs, which do not belong to the academic domain (e.g., ∼51% of the homepages in our dataset are not from the .edu domain). An alternative, more efficient approach, is to use a broader Web search for author discovery together with an accurate homepage classifier.

Furthermore, we propose a novel search-then-classify approach to find researchers homepages on the Web and identify them with powerful deep learning models. Our approach is inspired from the way humans search for scholarly information on the Web. Using hints from the titles, snippets and the URL strings, human searchers are often able to locate the correct homepage from the Web search results, e.g., by navigating from the paper URL to the index

of the homepage where the paper is located. To illustrate this process, Figure 3.2 shows an example of a Web search using the Bing search engine for the title of a paper published in WWW 2008. In the figure, the link shown in hosted on the homepage of the first author of the searched paper, while the links shown in red point to newly discovered authors. For the paper title search, the homepage of the first author of the searched paper can be accurately retrieved by navigating from the paper's URL to the index of the homepage (see the first result of the figure). Interestingly, notice that homepages belonging to seven other authors (different from the authors of the searched paper) can be discovered through the title search. We posit that this is because scientific paper titles comprise a large fraction of keywords[108], and hence, the words in paper titles serve as excellent keywords to formulate queries that can retrieve topically-related research papers, which are likely to be hosted on researchers' homepages.

Our search-then-classify approach specifically captures the above aspects by first "issuing" a query to find relevant content from the Web, and subsequently using a *homepage classification* module to identify homepages from the retrieved content (i.e., from the Web search results). Identifying homepages "in the wild" is very challenging since they have different structures and content and the URLs where they are hosted are very diverse and new URLs appear over time (e.g., many researchers use now github for their homepages, which was unlikely 5 or 10 years ago). Using author names and paper titles as queries, together with a homepage classifier, we are able to discover not only homepages of intended authors (e.g., those searched directly by name or those of the searched titles), but also homepages of other authors who work on semantically-related topics. We use deep learning models to learn powerful representations of URLs and page content. Moreover, we explore self training[109] in order to reduce the human effort needed to label the data by exploiting unlabeled data for the homepage classification task.

Our contributions are as follows:

- We propose a novel integrated framework based on search-driven methods to automatically acquire research documents for scientific collections. To our knowledge, we are

the first to use "Web Search" based on author names to obtain seed URLs for initiating crawls in an open-access digital library. Moreover, we design a traditional machine learning based novel homepage identification module and adapt existing research on academic document classification, which are crucial components of our framework. We show experimentally that our homepage identification module and the research paper classifier substantially outperform strong baselines.

- To automatically acquire research documents, we perform a large-scale, first-of-its-kind experiment using $43,496$ research paper titles and $32,816$ author names from Computer and Information Sciences. We compare our framework with two baselines, a breadth-first search crawler and, to the extent possible, the Microsoft Academic. We discuss that our framework does not substitute these systems, but rather they very well complement each other. We compare our framework with two baselines, a breadth-first search crawler and, to the extent possible, the Microsoft Academic. We discuss that our framework does not substitute these systems, but rather they very well complement each other. As part of our contributions, we will make all the constructed datasets available.

- We propose a search-driven homepage finding approach that uses author names and paper titles to find researcher homepages. To our knowledge, we are the first to use *"paper titles"* as queries to discover researcher homepages. Furthermore, we explore Convolutional Neural Networks (CNNs) for author homepage identification,[2] which is a crucial component in our approach. We conduct a thorough evaluation of the CNN models trained on both URLs and page content, and show significant improvements in performance over baselines and prior works. Furthermore, we show that self training can improve the performance of the classifier with the small amount of labeled data along with the unlabeled data.

- To discover researcher homepages, we perform a large-scale experiment using author names and paper titles from Computer Science as queries, and show the effectiveness

[2]We use author homepage classification or identification interchangeably.

of our approach in discovering a large number of homepages. Finally, as part of our contributions, all resulting datasets for author homepage identification and homepage discovery will be made available to further research in this area.[3] We show the development and deployment requirements of our proposed approach in CiteSeerX and the maintenance requirements.

3.2 Related Work

Work related to our proposed search-driven approaches follows in several categories, including scientific portals, standard supervised learning (based on feature engineering) and graph-based approaches, semi-supervised learning approaches that can leverage unlabeled data and complementary sets of features (or views), deep learning approaches to text classification and focused crawling approaches. We discuss the most relevant works in each of these categories in the remaining of this section.

Scientific Portals. There are many prior works that have focused on enhancing digital libraries content to better satisfy the needs of digital library users[110–112]. Several works studied the coverage in scientific portals such as Microsoft Academic, Google Scholar, Scopus and the Web of Science[113;114].

Supervised and Graph-based Approaches. Homepage finding and document classification are well-studied in information retrieval. The homepage finding track in TREC 2001 resulted in various machine learning systems for finding homepages[115–117]. Author/researcher homepage identification is a type of homepage finding task, which has been studied extensively in the context of digital libraries such as CiteSeer[105] and ArnetMiner[106]. Among works focusing specifically on researcher homepages, both Tang et al.[118] and Gollapalli et al.[28] treated homepage finding as a binary classification task and used various URL and webpage content features for classification. Ranking methods were also explored for homepage finding using the top terms obtained from topic models[119]. Qi and Davison[120] used HTML structure-based features and content-based features for classifying webpages. Wang

[3] https://www.cs.uic.edu/∼cornelia/datasets/homepage_discovery

and Oyama[117] studied the problem of collecting researcher homepages for Japanese websites using on-page and anchor text features. Ye et al.[121] also focused on finding high quality researcher homepages. Kang et al.[122] used the last name of a given author followed by a title of one of his/her publication as a query to a search engine to locate a publication list of the author.

Semi-supervised Learning Approaches. For a given webpage, both URL and the HTML content can be used for classifying the webpage. Multi-view learning is usually considered as maximizing the unanimity between different views[123;124]. Co-training (usually with two views) is a type of multi-view learning. Gollapalli et al.[28] proposed an algorithm for "learning a conforming pair of classifiers" that imitates co-training by using the URL and the HTML content as two different views. Jing et al.[125] addressed the webpage classification problem using a discriminant common space by learning a multi-view shared transformation in a semi-supervised way. Self training[109] uses labeled and unlabeled data usually with a single view to improve the classifier performance.

Deep Learning Approaches. Despite the effectiveness of feature engineering used in traditional machine learning, this is a labor intensive task and sometimes fails to extract all the discriminative information from the data[126]. Existing models for homepage classification/identification generally use hand-engineered features extracted from URLs and page content[118;127]. However, improved semantic representations can be obtained directly from the data using deep learning, and can help avoid problems related to feature engineering. For example, Kim[128] used Convolutional Neural Networks for representation learning for sentence classification and achieved remarkable results. The author used one convolutional/pooling layer (consisting of filters of three different sizes), together with a word embedding layer that encodes tokens in the input sequence and experimented with several variants of word embeddings, including fixed pre-trained vectors, or randomly initialized word vectors later tuned for a specific task. Zhao et al.[129] used neural network based homepage identifier to find a homepage of a given researcher within a set of HTML pages which are retrieved for a given researcher name as a query. In contrast, inspired from Kim[128], we use CNNs for representation learning for homepage classification and aim to identify homepages of

semantically-related authors (not just the targeted author) for a given query.

Focused Crawling. Focused crawling was introduced by Chakrabarti et al.[130] to deal with the information overload on the Web in order to build specialized collections focused on specific topics. Zhuang et al.[111] demonstrated the feasibility of using author homepages as alternative resources to collect research papers that are missing from an academic digital library. Garcia et al.[131] proposed a framework to gather publication list of different researchers by using author names as queries to a web search engine.

As opposed to previous approaches that used researcher names and their affiliations to locate a given researcher's homepage, we focus on researcher homepage discovery and propose an approach that mimics how people skim through the search results to discover homepages on the Web. In addition, we are the first to interleave various components of Web search, crawl, and document processing to build an efficient paper acquisition framework.

3.3 Proposed Frameworks and Approaches

We have proposed search based approaches to automatically augment the scientific document collections and accurate lists of researchers' homepages. In this section we discuss both.

3.3.1 Scientific Documents Discovery

Figure 3.3 shows the control flow paths of our proposed framework to obtain research papers and thus augment existing collections. In **Path 1**, paper titles are used as queries and the PDF documents resulting from each title search are classified with a paper classifier based on Random Forest. Author names comprise the queries for Web search in **Path 2**, the results of which are filtered by a homepage identification module trained using RankSVM. The predicted author homepages from **Path 2** serve as seed URLs for the crawler module that obtains all documents up to a depth 2 starting from each seed URL. The paper classification module is once again employed to retain only research papers. Note that we crawl only publicly available and downloadable documents those appear in the search responses of the

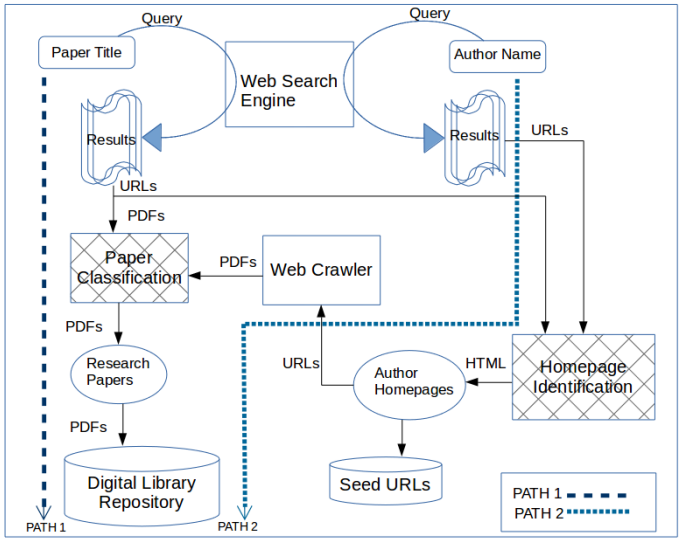


Figure 3.3: *Schematic Diagram of our Scientific Documents Discovery Framework.*

Web search or from the researcher homepages.

Homepage Ranking

Among the works focusing on researcher homepages, both Tang et al.[118] and Gollapalli et al.[127] treated homepage finding as a binary classification and used URL string features and content features (extracted from the entire .html page) for classification. However, given our Web search setting, the non-homepages retrieved in response to an author name query can be expected to be diverse with webpages ranging from commercial websites such as LinkedIn, social media websites such as Twitter and Facebook, and several more. To handle this aspect, we frame homepage identification as a supervised ranking problem. Thus, given a set of webpages in response to a query, our objective is to rank homepages higher relative to other types of webpages, capturing our preference among the retrieved webpages. Preference information needed for the ranking can be easily modeled through appropriate objective functions in learning to rank approaches[132]. For example, RankSVM[133] minimizes the Kendalls τ measure based on the preferential ordering information in the training examples. We design the following feature types for our ranking model, which capture aspects (e.g., snippets) useful for a Web user to find homepages:

1. **URL Features**: Intuitively, the URL strings of academic homepages can be expected to contain (or not) certain tokens. For example, a homepage URL is less likely to be hosted on domains such as "linkedin" and "facebook." On the other hand, terms such as "people" or "home" can be expected to occur in the URL strings of homepages (see examples of homepage URLs in Figure 3.1). We tokenize the URL strings based on the "slash (/)" separator and the domain-name part of the URL based on the "dot (.)" separator to extract our URL and DOMAIN feature dictionaries.

2. **Term Features**: The current-day search engines display the Web search results as a ranked list, where each webpage is indicated by its HTML title, the URL string as well as a brief summary of the content of the webpage (also known as the "snippet"). We posit that Scholarly Web users are able to identify homepages among the search results based on the term hints in titles and snippets (for example, "professor", "scientist", "student"), and use words from titles and snippets to extract our TITLE and SNIPPET dictionaries.

3. **Name-match Features**: These features capture the common observation that researchers tend to use parts of their names in the URL strings of their homepages[118;127]. We specify two types of match features: (1) a boolean feature that indicates whether any part of the author name matches a token in the URL string, and (2) a numeric feature that indicates the extent to which name tokens overlap with the (non-domain part of) URL string given by the fraction: $\frac{\#\texttt{matches}}{\#\texttt{nametokens}}$. For the example author name "Soumen Chakrabarti" and the URL string: `www.cse.iitb.ac.in/~soumen`, the two features have values "true" and 0.5, respectively.

The dictionary sizes for the above feature types based on our training datasets (see Section 3.4.1) are listed below:

Feature Type	Size
URL+DOMAIN term features	2025
TITLE term features	19190
SNIPPET term features	25280
NAME match features	2

Paper/Non-Paper Classification

In order to obtain accurate paper collections, it is important to employ a high-accuracy paper/non-paper classifier. Caragea et al.[100] studied the classification of academic documents into six classes: Books, Slides, Theses, Papers, CVs, and Others. The authors showed that a small set of 43 structural, text density, and layout features (Str) that are designed to incorporate aspects specific to research documents, are highly indicative of the class of an academic document. Because we are mainly interested in research papers to augment research collections and because binary tasks are considered easier to learn than multi-class tasks[134], we adapted this prior work on multi-class document type classification[100] and re-trained the classifiers for the two-class setting: paper/non-paper.

3.3.2 Researchers' Homapages Discovery

Task Description. Our task is to automatically discover new researchers' homepages on the Web, and augment and maintain up-to-date lists of homepages in open-source digital libraries to enable effective and efficient crawls for collecting documents. To address this task, we propose a search-then-classify approach for discovering researchers' homepages from the Web that mimics the search process adopted by humans. Author names and paper titles, freely available on the Web for specific subject disciplines are used to form suitable queries in our approach. Specifically, Path 1 starts with queries for authors names, while Path 2 starts with queries for paper titles. To identify researchers' homepages, pages retrieved on either path are classified with a CNN model. Table 3.1 shows examples of queries issued in Path 1 (author names) and Path 2 (paper titles), respectively.

Path 1: Author Name Query
Eric T. Baumgartner filetype:html
Path 2: Paper Title Query
Solving Time-Dependent Planning Problems. filetype:pdf

Table 3.1: *Example of author name and paper title queries.*

Application Description. Our implementation of the search-then-classify framework represents a critical part towards a sustainable CiteSeerX, in that it maintains and augments up-to-date lists of researchers' homepages found on the Web. Given the infeasibility of collecting the entire content on the Web, our search-then-classify framework aims to minimize the use of network bandwidth and hardware resources by selectively crawling only pages relevant to a (specified) set of topics.

Innovative Use of AI Technology

Convolutional Neural Networks. A key component in our framework is a classification module that identifies whether a retrieved webpage is a homepage or not. Inspired by Kim[128], we use Convolutional Neural Networks for representation learning of URLs and page content. Convolutional Neural Networks (CNNs)[135] are a special kind of neural networks to process grid-like structured data, including sequence or time series data. CNNs are associated with the idea of a "moving filter." Thus, in our approach, we explore a CNN based classifier for identifying homepages from the retrieved results in both search paths.

The CNN architecture used in our experiments is shown in Figure 3.4, and is comprised of mainly three layers: a word embedding layer, a convolutional layer followed by max pooling, and a fully connected layer for the classification. The embedding layer for tuning task specific word embeddings is initialized to a random vector corresponding to the input sequence. A convolutional layer consists of multiple filters of different sizes (e.g., sizes 3 and 4, respectively) that generate multiple feature maps (e.g., 300) for each filter size. Pooling is usually used after the convolutional layer to reduce the dimensionality (i.e., number of parameters) and prevent overfitting. The common practice for text is to extract the most important feature within each feature map[136], called 1-max pooling. In our architecture, 1-max pooling is applied over each feature map and the maximum values from each filter

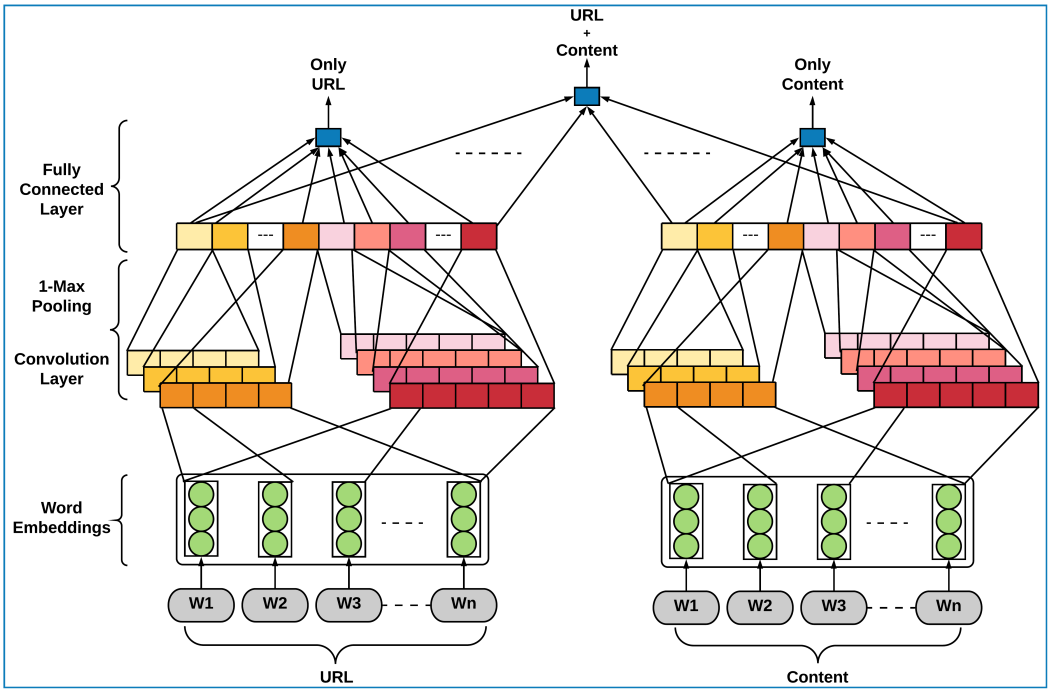


Figure 3.4: *Illustration of our CNN architecture used for homepage classification.*

are selected. These maximum values are then concatenated and used as input to a fully connected layer for the classification task (homepage versus not-homepage). We minimize a sigmoid (or binary) cross-entropy loss function using Adam optimizer to correctly predict the class label. If y_i is the true label and $p(y_i)$ is the predicted label, then the cross-entropy loss function (L) for N examples is calculated as:

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i.log(p(y_i)) + (1 - y_i).log(1 - p(y_i)) \tag{3.1}$$

We investigate two types of representations derived using CNN: (1) word based HTML content; and (2) word based URL. As can be seen from Figure 3.4, we explore the CNN models individually on either URL or page content, or jointly on both URL and page content. The URLs and corresponding pages (content) are those obtained as the result of our search for author name and paper title queries.

For the word based HTML model, we consider the first 1000 words from the HTML content of each page (given that the average length of HTMLs in the homepage class in our

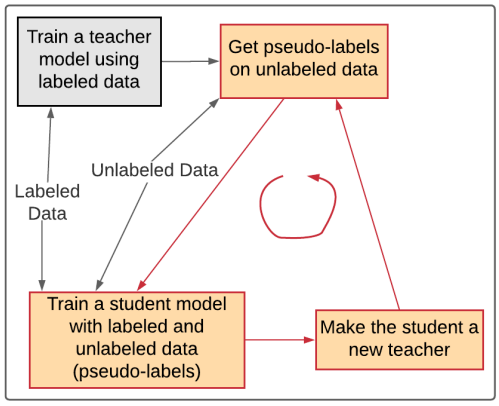


Figure 3.5: *Teacher-Student architecture of self training.*

dataset is 982, and most of the homepage characteristics often appear in the beginning of a page). Furthermore, we remove stop words and digits, and consider words appearing in at least 10 documents. For the URL based model, we tokenize the URLs with '/' as a delimiter, and form the vocabulary from all unigrams that appear in WordNet[137]. Consistent with Gollapalli et al.[28], for words that do not appear in WordNet, we add URL string patterns to the vocabulary, including underscored or hyphenated words, words with the '∼' sign, alphanumeric words, and long words (i.e., words with more than 30 characters). These patterns can help to filter out course pages, announcements, calendars, etc. For this model, we consider words appearing in at least 3 URLs.

Semi-supervised Teacher-Student Model. As discussed in the Introduction, one major challenge in identifying homepages is that the URLs where homepages can be hosted change over time. In order to reduce the human effort for data annotation, we investigate self-training in a Teacher-Student fashion to utilize the unlabeled data together with already labeled data. The model works in four steps in an iterative manner (Figure 3.5): (1) labeled data is used to train a teacher model; (2) the teacher model is used on unlabeled data to generate pseudo labels; (3) the student model is trained using both labeled data and pseudo labeled data (unlabeled data); (4) iterate the process by putting back the student as a teacher to generate new pseudo labels for training new student model. We considered examples which are predicted positive or negative with the probability ≥ 0.8 and ≤ 0.2, respectively. In our case, we used *data balancing* while using pseudo labeled data and sampled same number

Actual URL	www.cc.gatech.edu/∼mnaik7/pubs/popl16.pdf
Candidate URLs	www.cc.gatech.edu/∼mnaik7/pubs/ www.cc.gatech.edu/∼mnaik7/ www.cc.gatech.edu/

Table 3.2: *Example URLs and candidate URLs.*

of examples as the training set with equally sampling from each slot of 0.05 range. As an example, for the positively labeled data using the teacher model we sample equally from 0.8 to 0.85, 0.85 to 0.90, and so on. The backbone of our model is the CNN on both page content and URL.

Generating Candidate URLs

Note that for each query, a set of URLs are retrieved. We discard responses from a list of 25 domains such as "ResearchGate", "LinkedIn", etc. Candidate homepage URLs for each retrieved URL in Path 2 are generated by first splitting the URL on "/" and then removing the last part of the URL, iteratively, until the domain is reached. In the candidate set of URLs, we keep only the URLs for which we are able to obtain the corresponding HTML. Examples of candidate URLs for a paper title search is shown in Table 3.2. For Path 1, we use retrieved search results for the candidate URLs.

3.4 Datasets

In this section, we explain the datasets used by our frameworks and their components.

3.4.1 Scientific Documents Discovery

The datasets used in the evaluation of our Scientific Documents Discovery framework and its components are summarized in Table 3.3 and are described below:

DBLP Homepages. For evaluating homepage finding using author names, we use the researcher homepages from DBLP. In contrast to previous works that use this dataset to train

Dataset		
DBLP Homepages		42,548(T) 4,255(+)
Research Papers	(Train)	960(T) 472(+)
	(Test)	959(T) 461(+)
CiteSeer^x	43,496 (Titles), 32,816 (Authors)	

Table 3.3: *Summary of datasets. Total and positive instances are shown using (T) and (+), respectively.*

homepage classifiers on academic websites[127], in our Web search scenario, the non-homepages from the search results of an author name query need not be restricted to academic websites. Except the true homepage, all other webpages therefore correspond to negatives. We constructed the DBLP homepages dataset as follows: DBLP provided a set of author homepages along with the authors' names. Using these authors' names as queries, we perform Web search using Bing API and scan the top-10 results[138] in response to each query. If the true homepage provided by DBLP is listed among the top-10 search results, this URL and the others in the set of Web results are used as training instances. We were able to locate homepages for $4,255$ authors in the top-10 results for the author homepages listed in DBLP.

Research Papers. To evaluate the paper/non-paper classifier, we used two independent sets of ≈ 1000 documents each, randomly sampled from the crawl data of CiteSeer$^{\mathtt{x}}$, obtained from Caragea et al.[100]. These sets, called Train and Test, respectively, were manually labeled with six classes: Paper, Book, Thesis, Slides, Resume/CV, and Others. We transform the documents' labels as the binary labels, Paper/Non-paper.

CiteSeer^x. Our third dataset is compiled from the CiteSeer$^{\mathtt{x}}$ digital library. Specifically, we extracted research papers that were published in venues related to machine learning, data mining, information retrieval and computational linguistics. These venues along with the number of papers in each venue are listed in Table 3.4. Overall, we obtained a set of $43,496$ paper titles and $32,816$ authors (unique names) for the evaluation of our framework at a large scale.

Total # of papers: 43,496, #authors (unique): 32,816
NIPS (5211), IJCAI (4721), ICRA (3883), ICML (2979), ACL (2970), VLDB (2594), CVPR (2373), AAAI (2201), CHI (2030), COLING (1933), KDD (1595), SIGIR (1454), WWW (1451), CIKM (1408), SAC (1191), LREC (1128), SDM (1111), EMNLP (920), ICDM (891), EACL (760), HLT-NAACL (692)

Table 3.4: *Conference venue (#papers) in the CiteSeer^{x} dataset.*

3.4.2 Researchers' Homepages Discovery

Here we discuss the datasets used in the evaluation of our Researchers' Homepages Discovery framework and its components.

DBLP Dataset

The WebKB dataset collected in 1997 has been previously used for homepage classification[28]. However, due to continuous changes in the information content on academic websites, this dataset has become outdated. For example, academic websites today contain invited talks, newsletters, job postings, and other events that do not occur in WebKB. In addition, the WebKB dataset does not contain homepages from industry lab researchers. To address the above limitations and in order to enable the exploration of deep learning for author homepage identification, we constructed a labeled dataset for this task as follows.

We obtained a list of author names and their homepages from DBLP in 2015. After data processing, we found that the original DBLP list had a repetition of 21 homepages and contained some URLs that were easily identified as non-homepages, specifically, 56 URLs from Wikipedia, 2 from DBLP and 1 from Springer. After removing repetitions and pages linking to Wikipedia, DBLP, and Springer, we ended up with a list of $13,239$ homepages. We further refined the original DBLP list by removing all outdated URLs, $5,596$ in total (HTTP 4xx client errors, non-valid homepages, or redirects to the default university/company page). We ended up with a list of $7,643$ author names and their homepages. We used the author names as queries to the Bing search API and retrieved the top-10 results for each query,

obtaining 76,375 search responses from Bing in total. From the Bing responses, we filtered out pages from the list of 25 domains such as "ResearchGate," "LinkedIn," "Wikipedia," "YouTube", etc. After removing such pages, and the overlap with the original DBLP set of homepages, we ended up with 20,229 Bing responses. We manually inspected the set of 27,872 URLs (7,643 DBLP URLs + 20,229 Bing URLs) for the labeling task, using three Amazon Mechanical Turk (AMT) workers and two undergraduate students, who were trained in an iterative fashion and worked closely with the researchers. Whenever there was agreement between the three AMT workers, we labeled the example accordingly (as homepage or non-homepage). When there was disagreement, the data was labeled further by the undergraduate students, and if a decision could not be reached, the final adjudication was made by one of the researchers. During the labeling task, we removed outdated and non-English pages. At the end of the annotation task, we found 8,529 positive examples (homepages) and 16,245 negative examples (non-homepages). After this manual annotation, we found that only 5,851 out of 7,643 DBLP homepages are valid (with the others being, e.g., moved to a different page). The result of this search validates our intuition that we can obtain homepages for the intended authors, but also additional homepages of related authors (e.g., co-authors).

In the set of negative domains, we observed that there were hundreds of URLs from domains such as "healthgrades.com", "ratemyprofessor.com", etc. Thus, we constructed the final dataset by sampling only 50 negative URLs from a given domain (Threshold $\theta \geq 50$). For domains with less than 50 URLs, we used all URLs. The final dataset used in our experiments contains 18,733 examples, specifically 8,529 homepages and 10,204 non-homepages.

Characteristics of the DBLP Dataset. Table 3.5 contains the characteristics of the DBLP dataset. As we can see, the percentage of URLs containing a $\sim$ sign or being from the '.edu' domain is higher in the positive set as compared to the two negative sets (specifically, 3,974 out of 8,529 URLs from the positive set contain the $\sim$ sign). On the other hand, in both negative sets, the percentage of URLs containing a digit or from the '.com' domain is higher as compared with the positive set. Moreover, 43%, 20%, and 23% of

	+ve	-ve ($\theta \geq 50$)	-ve (all)
#Examples	8,529	10,204	16,245
#URLs as a domain	439	1,097	1,156
#URLs with $\sim$ sign	3,974	113	120
#URLs from '.edu'	4,192	1,304	2,307
#URLs from '.com'	464	5,290	8,929
#URLs containing digit	1,034	5,191	9,130
#Pages with 'homepage' word or its synonyms	3,639	2,048	3,759
Max. #characters/URL	158	232	297
Avg. #characters/URL	32	50	52
Max. #words/webpage	86K	530K	530K
Avg. #words/webpage	982	2,011	2,418

Table 3.5: *Datasets characteristics.*

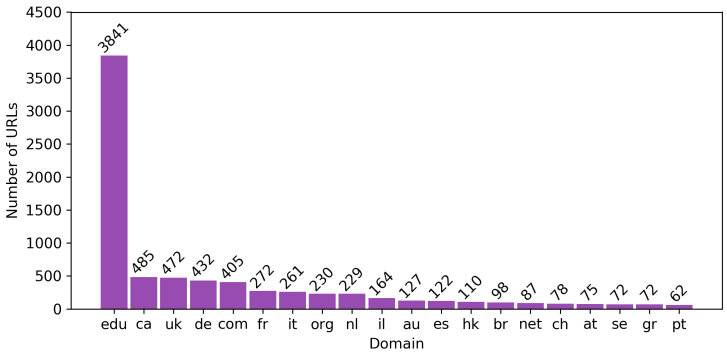


Figure 3.6: *Number of URLs corresponding to homepage from different domains in our DBLP dataset.*

webpages contain the keyword 'homepage' or its synonyms such as personal page, personal site, personal website, etc. in the positive set, negative set (threshold-50), and negative set (all), respectively. Also the maximum and average #characters per URL and #words per webpage are smaller in the positive set as compared with both negative sets. Figure 3.6 shows the top-20 domains for the positive set.

CiteSeerX Dataset

Our second dataset, which we used for the large scale evaluation of our overall framework, is compiled from CiteSeerX. Specifically, we extracted papers published in venues related to machine learning, information retrieval, and computational linguistics. Overall, we obtained a random set of $10,000$ paper titles and $14,808$ authors (unique names corresponding to

the selected titles) for the evaluation of our search-driven approach on a large scale. These venues along with the number of papers in each venue are as follows: ACL (1193), IJCAI (1167), COLING (1010), ICRA (827), NIPS (650), VLDB (613), ICML (564), AAAI (411), CHI (399), CVPR (371), KDD (366), EACL (333), SIGIR (305), SAC (296), SDM (242), ICDM (236), CIKM (235), WWW (231), LREC (226), HLT-NAACL (209), and EMNLP (116).

3.5 Experiments and Results

In this section, we describe our experiments on both frameworks, Scientific Documents Discovery and Researchers' Homepages Discovery, and their corresponding components.

3.5.1 Scientific Documents Discovery

In this section, we describe our experiments on homepage identification and paper classification along with their performance within the search then crawl then process paper acquisition framework.

Performance measures. We use the standard measures Precision, Recall, and F1 for summarizing the results of author homepage identification and paper classification[139]. Unlike classification where we consider the true and predicted labels for each instance (webpage), in RankSVM the prediction is per query[133]. That is, the results with respect to a query are assigned ranks based on scores from the RankSVM and the result at rank-1 is chosen as the predicted homepage.

Author Homepage Identification

We aim to determine how accurate is RankSVM in identifying a homepage for each author name query. Table 3.6 shows the five-fold cross-validation performance of the homepage identification on the positive class trained using RankSVM compared with various classification algorithms, Naïve Bayes, Maximum Entropy and Support Vector Machines. The results

in the table are averaged across all five test sets of cross-validation. Hyperparameter tuning (e.g., C for SVM) was performed on a development set extracted from training.

Method	Precision	Recall	F1
RankSVM	**0.8933**	0.8933	**0.8933**
Naïve Bayes	0.4830	**0.9239**	0.63432
MaxEnt	0.8207	0.8002	0.8102
Binary SVM	0.8353	0.8149	0.8249

Table 3.6: *RankSVM vs. supervised classifiers on DBLP.*

As can be seen from the table, RankSVM performs much better compared with the classification approaches, in terms of Precision and F1, although Recall is higher for Naïve Bayes. Hence, RankSVM is able to capture the relative preferential ordering among the search results and performs the best in identifying the correct author homepage in response to a query. A possible reason for the lower performance of the classification approaches such as Binary SVMs, Naïve Bayes, and Maximum Entropy is that they model the positive and negative instances independently and not in relation to one another for a given query. Moreover, the diversity in webpages among the negative class is ignored and they are modeled uniformly as a single class in the classification approaches.

Research Paper Classification

We compare the performance of classifiers trained using the 43 structural features (Str) with that of classifiers trained using the "bag of words" (BoW), URL-based features (URL), and a Convolutional Neural Network (CNN) model. For BoW and URL, we used the same text processing operations as in Caragea et al.[100]. We experimented with several classifiers: Random Forest (RF), Decision Trees (DT), Naïve Bayes Multinomial (NBM), and Support Vector Machines with a linear kernel (SVM). For CNN, we use the words as a sequence as an input; we first get the word embeddings as a part of the network followed by the CNN filter, max-pooling, concatenation, and the fully connected layer for the classification task, similar to Kim[128]. All models are trained on the "Train" dataset and are evaluated on the "Test" dataset. We tuned model hyper-parameters in 10-fold cross-validation experiments

on "Train" (e.g., C for SVM and the number of trees for RF).

Feature/Cls. (Setting)	Precision	Recall	F1
BoW / DT (P-B)	0.860	0.920	0.889
URL / SVM (P-B)	0.729	0.729	0.729
Str / RF (P-B)	**0.933**	**0.967**	**0.950**
CNN (P-B)	0.816	0.890	0.851
Str / RF (A-B)	**0.952**	**0.951**	**0.951**
Str / RF (P-M)	0.918	0.965	0.941
Str / RF (A-M)	0.893	0.902	0.892

Table 3.7: *Performance of paper classifier on "Test". "P" stands for the paper class, while "A" for the average of classes. "B" and "M" stand for binary and multi-class, respectively.*

Table 3.7 shows the performance (Precision, Recall, and F1) for the binary setting on "Test" for each feature type, BoW, URL, and Str, and the CNN, with the classifiers that give the best results for the corresponding feature type or model (first four lines). The results are shown for the "paper" class (P). In the table, we also show the performance on the "paper" class with the multi-class (M) setting and the weighted averages (A) of all measures over all classes for both the settings. As can be seen from the table, the best classification performance is obtained using Random Forest trained on the 43 structural features with the overall performance above 95% being substantially higher in the binary setting compared with the multi-class setting. The reason behind lower performance of the CNN classifier can be the wide variety of documents present in the dataset and the small number of the training examples.

Large-Scale Experiments

Finally, we evaluate our "search then crawl then process" framework and its components in practice in large scale experiments, using our CiteSeer$^{\text{x}}$ subset. To this end, we evaluate the capability of our framework to obtain large document collections, quantified by the number of research papers it acquires (through both paths). For **Path 1**, we use the 43,496 paper titles directly as search queries. Structural features extracted from the resulting PDF documents of each search are used to identify research papers with our paper classifier. For **Path 2**, the 32,816 unique author names are used as queries. The RankSVM-predicted homepages from

Title Queries
Knowledge-based Knowledge Elicitation. filetype:pdf
Solving Time-Dependent Planning Problems. filetype:pdf
Author Name Queries
Eric T. Baumgartner filetype:html
Nelson Alves filetype:html

Table 3.8: *Example of title and author name queries.*

the results of each author name query are crawled for PDF documents up to a depth of 2, using the wget utility.[4] Again, the paper classifier is employed to identify the papers from the crawled documents. In all experiments, we used the Bing API to perform Web searches. Examples of title and author name queries are provided in Table 3.8.

Overall Yield. The total numbers of PDFs and research papers found through the two paths in our Search/Crawl/Process framework are shown in Table 3.9 (the columns labeled as #CrawledPDFs and #PredictedPapers, respectively). Intuitively, the overall yield can be expected to be higher through **Path 2**. This is because once an author homepage is reached, other research papers that are linked from this homepage can be directly obtained. Indeed, as shown in the table, the numbers of PDFs as well as predicted papers are significantly higher along **Path 2**. Crawling the RankSVM-predicted homepages of the 32,816 authors, we obtain on average ≈ 14 research papers per query ($\frac{452273}{32816} = 13.78$). In contrast, examining only the top-10 search results along **Path 1**, we obtain ≈ 5 papers per query on average ($\frac{213683}{43496} = 4.91$). The high percentage of papers found along **Path 2** is consistent with previous findings that researchers tend to link to their papers via their homepages[127;140]. Note that in all experiments, since the original 43,496 titles are extracted from CiteSeer$^{\mathbf{x}}$, for a fair evaluation, we removed all title search results that point to the CiteSeer$^{\mathbf{x}}$ domain, i.e., http://citeseerx.ist.psu.edu/.

Furthermore, the numbers of unique papers found along each of the two paths are shown in Table 3.9 (the column labeled as #UniquePapers). We used ParsCit[5] to extract the titles of the research papers obtained from both the paths and then calculated the duplicates from

[4] https://www.gnu.org/software/wget/
[5] http://aye.comp.nus.edu.sg/parsCit/

#Queries	#CrawledPDFs	#PredictedPapers	#UniquePapers	#MatchesWithOriginalTitles
43,496 titles (Path 1) 32,816 names (Path 2)	322,029 665,661	213,683 452,273	91,237 204,014	32,565 17,627
Overlap: **Path 1** & **Path 2**	-	-	**28,374**	**16,188**
Total # of papers: **Path 1** + **Path 2**	-	-	**266,877**	**34,004**

Table 3.9: *Number of papers obtained through* **Path 1** *and* **Path 2** *in our Search/Crawl/Process framework.*

these titles.[6] As can be seen from the table, we are able to obtain 91,237 and 204,014 unique papers from **Path 1** and **Path 2**, respectively, which account for ≈ 2 papers per title query on average ($\frac{91237}{43496} = 2.09$) and ≈ 6 papers per author query on average ($\frac{204014}{32816} = 6.21$). However, since our objective is not to use one path or the other, but use a combination of both Path 1 and Path 2, we further expanded our analysis to show the overlap between Path 1 and Path 2 in terms of unique titles.

Overlap between Path 1 and Path 2. Table 3.9 shows also the overlap in the two sets of unique papers (between **Path 1** and **Path 2**), which is 28,374. Compared to the overall yields along **Path 1** and **Path 2** (213,683 and 452,273, respectively) and even with the number of unique papers along each path, this small overlap indicates that the two paths are capable of reaching different sections of the Web and play complementary roles in our framework. For example, the top-20 domains of the URLs from which we obtained research papers along **Path 1** are shown in Figure 3.7. As can be seen from the figure, via Web search, we are able to reach a wide range of domains. This is unlikely in crawl-driven methods without an exhaustive list of seeds since only links up to a specified depth from a given seed are explored[139]. Interestingly, using a combination of both Path 1 and Path 2, we were able to obtain 266,877 (=91,237+204,014-28,374) unique papers.

Next, we investigate the recovery power of our framework. Precisely, how many of the original 43,496 titles were found through each path as well as their combination?

Overlap with the Original Titles. The numbers of papers that we were able to obtain from the original 43,496 titles through both paths are shown in the last column of

[6]To find duplicates, we convert the text to lowercase, and remove punctuation and whitespace.

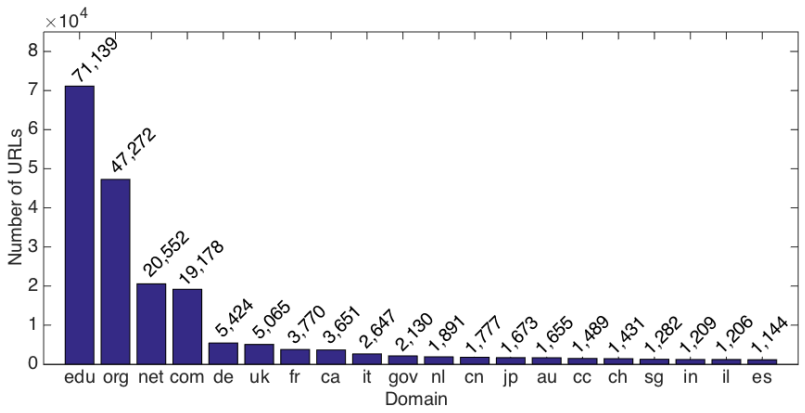


Figure 3.7: *The top-20 domains from which papers were obtained along Path 1 of our framework.*

Table 3.9, labeled as #MatchesWithOriginalTitles. To compute these matches, we used the title and author names available in our CiteSeer^{x} subset to look up the first page of each PDF document. As can be seen from the table, we were able to recover 75% ($\frac{32565}{43496}$) of the original titles through **Path 1** compared to the 40% ($\frac{17627}{43496}$) through **Path 2**. The total number of matches with the original titles between **Path 1** and **Path 2** was 16,188. Overall, through a combination of both **Path 1** and **Path 2**, we were able to recover 78% ($\frac{34,004}{43496}$) of the original titles (34,004=32,565+17,627-16,188 papers obtained through both paths out of the original titles).

To summarize, using about 76,312 queries ($43,496 + 32,816$) through **Path 1** and **Path 2**, we are able to build a collection of 665,956 papers ($213,683 + 452,273$) and 266,877 unique titles ($91,237 + 204,014 - 28,374$). About 32-33% of the obtained documents are "non-papers" along both paths. Scholarly Web is known to contain a variety of documents including resumes, and presentation slides[141]. Some of these documents may include the exact paper titles and may appear in paper search results as well as be linked from author homepages.

Anecdotal Evidence. Given the size of our CiteSeer^{x} dataset and the large number of documents obtained via our framework (as shown in Table 3.9), it is extremely labor-intensive to manually examine all documents resulting from the large scale experiment. However, since our classifiers and rankers achieve performance above 95% and 89% based on

our test datasets compiled specifically for these tasks, we expect them to continue to perform well "in the wild." We show anecdotal evidence to support this claim, i.e., an estimate of how many true papers we are able to obtain via our Search/Crawl/Process framework starting from a small set of titles.

To obtain such an estimate, we randomly selected 10 titles from the CiteSeer^{x} dataset. From the corresponding papers of these 10 titles, we extracted 33 unique authors. We manually inspected all PDFs that can be obtained via title search (**Path 1**) as well as the homepages obtained via author name search (**in Path 2**) That is, through **Path 1**, we searched the Web for the 10 selected titles and manually examined and annotated the top-10 resulting PDFs for each title query. The title search resulted in 59 PDFs, of which 33 are true papers and 26 are non-papers. Our paper classifier predicted 32 out of 33 papers correctly and 38 papers overall and achieved a precision and recall of 84% and 97%, respectively.

Similarly, through **Path 2**, we searched for the 33 author names from the Web and manually examined and annotated the top-10 resulting webpages for each author name query. From the author search, manually, we were able to locate 19 correct homepages of the 33 authors. A manual inspection of the predicted homepages revealed that our framework was not able to locate 6 out of the 19 correct homepages. Table 3.10 shows a few examples where our framework was not able to locate the correct homepages. For example, occasionally, RankSVM ranks university faculty profile or faculty research group at the first rank, which is then predicted as a homepage by RankSVM (e.g., URLs 4 and 6 in Table 3.10). URL 5 in Table 3.10 is wrongly predicted by RankSVM as the homepage for the researcher name "David Bell." This is precisely because there is a well known novel writer and also baseball player with the same name, which get ranked higher in the results of the search engine. Note that, interestingly, the actual homepage of David Bell, corresponding to URL 2 was not retrieved in the top 10 search responses of Bing.

Baseline Comparisons

Breadth-first search crawler. We compare our Search/Crawl/Process framework, through

Actual homepage
1. http://destrin.smalldata.io/ 2. http://www.cs.qub.ac.uk/∼D.Bell/dbell.html 3. http://www.ai.sri.com/∼yang
Predicted homepage
4. http://research.cens.ucla.edu/people/estrin/ 5. http://davidbellnovels.com/ 6. http://www.ai.sri.com/people/yang/

Table 3.10: *A few examples where our framework was not able to locate the correct homepage*

Path 1, with a breadth-first search crawler as implemented in CiteSeer^{x}. The CiteSeer^{x} crawler starts with a list of seed URLs, performs a breadth-first search crawl and saves open-access PDF documents.

For this experiment, we randomly selected $1,000$ titles from DBLP. We then searched the Web for these titles and retrieved the top-10 resulting PDFs for each query. Through this search, we obtained a total of $5,793$ PDFs, from which we removed 110 documents that were downloaded from CiteSeer^{x}, since they were obtained as a result of the CiteSeer^{x} breadth-first search crawler. Note that there is an overlap of 6 documents, i.e., only located on CiteSeer^{x} (and nowhere else on the Web), between the 110 removed documents and the 1000 DBLP initial titles. From the remaining documents, our paper classifier predicted $3,427$ documents as papers, out of which $2,797$ are unique papers/titles. We searched CiteSeer^{x} for these $2,797$ titles to determine how many of them are found by the CiteSeer^{x} crawler. We found $1,037$ titles in CiteSeer^{x} by checking if one title string contains the other. Thus, with our framework, we were able to obtain $2,797 - 1,037 = 1,760$ additional papers. Out of the 994 $(1000 - 6)$ DBLP titles, only 121 papers were found by both our framework and the CiteSeer^{x} crawler. In addition, our framework found 165 more papers (with a total of 286 out of 994 DBLP titles), whereas the CiteSeer^{x} crawler found only 92 more papers (with a total of 213 out of 994 DBLP titles). Moreover, out of the additional yield of our framework, i.e., 2511 $(= 2797 - 286)$ papers, only 552 are found by the CiteSeer^{x} crawler (identified by searching for the 2511 titles in the CiteSeer^{x} digital library - by exact match). These results are summarized in Figure 3.8. We note that the two approaches are not substituting, but rather complementing each other.

Figure 3.8: *Comparison of the Search/Crawl framework with the CiteSeer^x breadth-first search crawler.*

Microsoft Academic. Searching for feeds from publishers (e.g., ACM and IEEE) and using webpages indexed by Bing is also considered by Microsoft Academic (MA) to collect entities such as paper, author, and venue, to be added to the MA graph[96]. An edge in the graph is added between two entities if there is a relationship between them, e.g., *publishedIn*. In contrast, in our framework, we collect not only the intended paper for a title search, but also all papers that are found for that search. In addition, we identify author homepages through author name search and, unlike MA, we use them to collect research papers from these homepages. To our knowledge, we are the first to use "Web Search" based on author names to obtain seed URLs for initiating crawls to acquire documents in scientific portals. Both our framework and MA use Bing for searches. Thus, using MA strategy to collect paper entities, 32,565 papers are recovered out of the 43,496 original titles. Adding the author search in our framework, we are able to collect an additional 1,439 (=34,004 − 32,565) papers from the original titles and 234,312 (=266,877 − 32,565) overall additional unique papers (see Table 3.9).

3.5.2 Researchers' Homepages Discovery

Next, we describe our experiments and results on homepage classification along with the performance of our overall search-then-classify framework.

Model	Precision	Recall	F1	Accuracy
CNN-URL	**0.91**	0.75	0.82	85.11%
CNN-Content	0.89	0.90	0.89	90.38%
CNN-Combined	0.90	**0.94**	**0.92**	**92.35%**
Co-training	0.87	0.86	0.87	87.64%
RF-URL	0.89	0.84	0.87	88.10%
RF-Content	0.83	0.92	0.87	87.35%
RF-Combined	0.85	0.91	0.88	88.55%

Table 3.11: *CNN vs. co-training and supervised models.*

Author Homepage Classification

To evaluate the performance of the CNN models on homepage classification, and compare them with previous approaches for this task, we divided our DBLP dataset into train, validation and test sets. The train, validation, and test sets have 60%, 20% and 20% examples, respectively. All the splits are constructed by keeping the original distribution of the URL set. We use the validation set for parameter tuning and model selection. We report precision, recall and F1-score for the positive class and the overall accuracy for each model on the test set (using the model that performed best on the validation set). For CNN models, we run the experiments with three different random initialization of the network weights and we report average values for each measure. The model that achieved the highest performance on the test set was chosen for the large scale experiment.

CNN vs. Supervised Models and Co-training. We contrast the performance of CNNs on different input types (URL, HTML content, and the combination) with the performance of several traditional supervised classifiers (Random Forest, Decision Trees, Naïve Bayes, and Support Vector Machines with a linear kernel), as well as with the performance of a semi-supervised co-training classifier for homepage identification as described in[28].

We used the *tf-idf* vector representations for all input types (URL and content based) for the traditional supervised classifiers. We trained the CNN models using mini-batches of size 64, with a sigmoid cross-entropy loss function and Adam optimizer with a learning rate of 0.0005. After experimenting with a large spectrum of parameters for CNNs on the validation set, the best parameters are as follows: for CNN-URL, 100 embedding size and 100 filters of size 5; for CNN-content, 300 embedding size and 100 filters of size 5. For co-training, we

obtained the code and unlabeled data from Gollapalli et al. [28] .[7]

Table 3.11 shows the performance on the test set of the CNN classifiers compared with co-training and supervised Random Forest (RF) classifiers. The RF classifiers performed the best among all the traditional supervised classifiers (and therefore we only show its results in the table). CNN classifiers are comparable and in many cases outperform their traditional supervised counterparts except the URL based classifier, and also the co-training approach proposed in[28]. We can also see that the CNN-combined, which uses both word-based content and word-based URL, achieves the highest performance among all models, in terms of recall and F1-score. For example, CNN-combined achieves the highest F1 of 0.92, whereas the RF-combined (URL + content) yields an F1-score of 0.88. We can also observe that CNN-URL achieves a highest precision of 0.91.

The Effect of Self-training. To see the effectiveness of self-training, we perform experiments using different portion of the training data along with unlabeled data to train our homepage classifier (CNN-Combined) using self-training. For the unlabeled data used in the self-training of our homepage classifier, we use candidate URLs of Path 1 (author name queries) of our proposed framework. We used $56,339$ HTML pages and URLs as the unlabeled set. We went up to 5 iterations of self-training, iteratively learning a teacher-student model. Table 3.12 shows the results highlighting the effect of self-training on the CNN-Combined classifier. We can see that the performance of the classifier improves when the labeled data is $\leq 25\%$ (only $2,811$ labeled examples). For example, while using only 1% (112) labeled examples, the performance of the classifier increased by $\approx 8\%$ using self-training from F1 of 0.78 to 0.84. For the error analysis and the large-scale experiments, we used the CNN-Combined classifier trained using self-training with 100% training examples. These results show that self-training can be very useful when we want to deploy/train the homepage classifier for other domains where the labeled data is not easily available, but we can easily collect the unlabeled data relevant to the task. Also, self-training can be very useful in order to reduce the human labeling effort, especially given the changing nature of the URL types/domains where homepages can be located.

[7]https://sites.google.com/site/sujathadas/home/datasets

Labeled Data	Precision	Recall	F1	Accuracy
Without self training				
1%	0.77	0.80	0.78	79.37%
5%	0.88	0.86	0.87	88.33%
10%	0.88	0.89	0.88	89.20%
25%	0.88	0.92	0.90	90.81%
50%	0.89	**0.94**	**0.92**	92.06%
100%	0.90	**0.94**	**0.92**	92.35%
With self training (uses unlabeled data)				
1%	0.83	0.85	0.84	84.96%
5%	0.85	0.91	0.88	88.85%
10%	0.87	0.91	0.89	89.60%
25%	0.87	**0.94**	0.91	91.06%
50%	0.90	0.93	**0.92**	92.14%
100%	**0.91**	0.93	**0.92**	**92.63%**

Table 3.12: *The performance of CNN-Combined model with and without self training.*

Error analysis. Table 3.13 shows sampled URLs along with model's confidence value where CNN-Combined model made errors. The blue part of each URL indicates the part of the URL that is used as an input to a deep learning model. Most of the false positive examples are pointing to webpages of a research group/lab or their list of people/members, and to webpages containing basic information regarding a professor or a person. Specifically, URL-1, URL-2 and URL-3 are pointing to a webpage of a member of the research group/lab, while URL-4 is pointing to a webpage containing basic information regarding a professor. Most of the false negative examples are pointing to a homepage containing very little research-related information, or a homepage with very different content than that expected on a usual researcher homepage. URL-5 contains very little HTML content (unlike a typical researcher homepage). URLs 6, 7, and 8 are also homepages with very different content than the content on a usual researcher homepage.

Large-Scale Experiments

We now evaluate the capability of our search-then-classify approach to discover author homepages in a large-scale experiment, using the CiteSeerX dataset. To this end, we use the 14,808 author names and the 10,000 paper titles as search queries on the Bing search API

False Positives	
Example	Confidence
1. http://users.ics.aalto.fi/eugenc	1.0
2. http://www.nott.ac.uk/∼itzbl/	1.0
3. http://www.eng.usf.edu/∼rfrisina/	0.9999
4. http://www.ssrc.ucsc.edu/person/phartel.html	0.9952
False Negatives	
Example	Confidence
5. http://zh-anse.com	0.9995
6. http://www.brookings.edu/experts/yuq	0.9989
7. https://blog.xot.nl/about-2	0.9858
8. http://freudenbergs.de/bert	0.9805

Table 3.13: *Errors made by CNN-Combined model, along with model's confidence values. The blue part in each URL indicates the part of the URL that is used as input to a model.*

Queries	**URLs**	**Candidate URLs**	CNN-comb. HPs	
			All	**Unique**
Author names	148,042	56,339	12,093	11,016
Paper titles	75,612	51,451	17,685	12,199
Overlapping	-	-	-	2,622
Total	-	-	-	20,593

Table 3.14: *Homepages from* $10,000$ *title and* $14,808$ *author search responses in a large-scale experiment.*

and employ the CNN-combined homepage classifier to identify homepages from the top-10 search results of each query. We used only the top-10 results as they are shown to often be sufficient to retrieve the relevant information [138].

Overall yield. The total number of Bing URL responses for our author name and paper title queries, the number of resulting candidate URLs (corresponding to the retrieved URLs), and the number of predicted homepages (overall and unique) obtained using the CNN-combined classifier are shown in Table 3.14. Individually, for the $14,808$ author name queries we obtained $148,042$ Bing URL responses (i.e., URLs pointing to html pages). After filtering out the easy-to-identify non-homepage commercial URLs, we generated $56,339$ candidate URLs (see Table 3.2 for examples of candidate URLs), out of which $12,093$ are classified as homepages by our CNN-combined classifier. From this set of $12,093$ predicted homepages, we obtained $11,016$ unique author homepages.

For the 10,000 paper title queries, we obtained 75,612 Bing URL responses (i.e., URLs pointing to PDF documents). After filtering out the commercial URLs, we generated 51,451 candidate URLs, out of which 17,685 are classified as homepages by our CNN-combined classifier. From this set of 17,685 predicted homepages, we obtained 12,199 unique author homepages.

Table 3.14 shows also the overlap in the two sets of unique homepages between author name search and paper title search, which consists of 2,622 unique homepages. As expected, this small overlap of homepages between author name search and paper title search indicates that research paper titles, formulated as queries, have a great potential to discover new researcher homepages (in addition to the homepages of researchers searched specifically). Precisely, through the paper title search and using the CNN-combined classifier, we were able to find an additional 9,577 (= 12,199−2,622) unique homepages (as compared to the author name search). This result also suggests that the two types of searches complement each other and are capable to reach different sections of the Web. The total number of homepages we discover through the author name search and paper title search in our approach is 20,593 (= 11,016 + 12,199 - 2,622). This total number of homepages showcases the potential of our approach to acquire and maintain large collections of homepages.

To see where the homepages acquired using our approach come from (what parts of the Web), we extracted the number of different domains and ranked them based on the number of URLs in each domain. Figure 3.9 shows the top-20 domains for the 20,593 homepages. In total, we found 107 domains. Around 48% of homepages are from the ".edu" domain. The top-20 domains shown in the figure cover ≈ 89% of total homepages that we discovered.

To see for how many authors out of 14,808 authors we were able to locate the homepages, we used Stanford Named Entity Recognizer. [8] We used the first appearing name from the predicted homepage as the owner of the homepage. To match the author names, we consider a match if we were able to match first and last name. We recovered homepages for 5,815 and 2,782 intended authors using author names and paper title queries, respectively. These results also showcase the potential of paper title queries to locate the homepages of other

[8] https://nlp.stanford.edu/software/CRF-NER.html

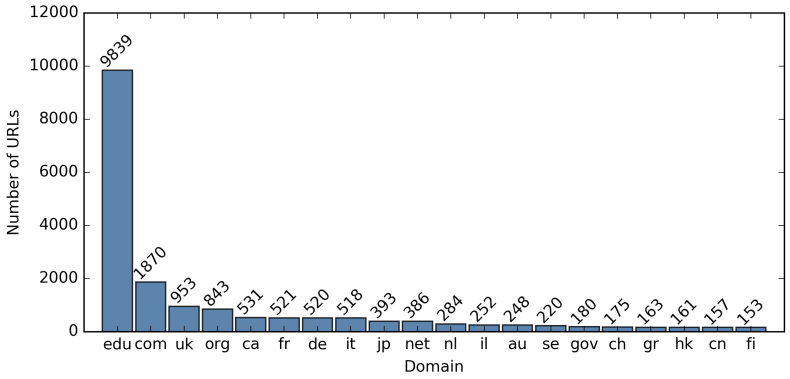


Figure 3.9: *Top-20 domains from author and title searches.*

authors than the authors of the queried titles.

Overlap with DBLP author homepages. One interesting question that can be raised is the following: "Is our approach able to discover homepages that are not already available in some online resource?" That is, starting with our sets of author names and paper titles, which are independent from the author names in the DBLP list of homepages, how many homepages can we discover that are not already in the DBLP list? To answer this question, we compare our overall $15,203$ homepages predicted by the CNN-combined classifier with the list of known DBLP homepages ($5,851$ homepages). The overlap between the $15,203$ predicted homepages in our approach and $5,851$ DBLP homepages is $1,882$. Hence, remarkably, overall, we are able to discover $18,711 = 20,593$ - $1,882$ homepages that are not present in our DBLP dataset.

Human Assessment and Validation. Key to our large scale experiment is to ensure data quality of the discovered homepages. In order to analyze the impact of the system effectiveness, we performed human assessment and validation. Specifically, we sampled 600 CNN-predicted homepages for human assessment and validation. We asked a human annotator (the first author of this paper) to determine if each page provided in the set is a homepage or not. The human annotator labeled 496 out of 600 URLs as homepages. In other words, 82.67% URLs from the sampled set were identified as true homepage. Close inspection of the remaining 104 URLs revealed that, most of those URLs are pointing to a group or lab or course page. Furthermore, we noticed that 266 out of 600 URLs contain a

$\sim$ sign, and 246 URLs with a $\sim$ sign were marked as a homepage by the human annotator.

3.6 Development and Deployment of Researchers' Homepages Discovery Framework in CiteSeerX

Although CiteSeerX utilizes open source software packages, many core components are not directly available from open source repositories and require extensive programming and testing. The current CiteSeerX codebase inherited little from its predecessor's (CiteSeer) for stability and consistency. The core part of the main web apps were written by Dr. Isaac Councill and Juan Pablo Fernández-Ramírez and many components were developed by other graduate students, postdocs and software engineers, which took at least 3-4 years. Different components of CiteSeerX are built using several languages such as JAVA, Python, Perl, Scala, etc. The homepage classifier component is developed using Python 2.7. We have used the Amazon AWS service for training the deep learning-based homepage classifier. The AWS instance that was used for training the classifier has Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz processor, 64GB RAM, and Tesla V100 SXM2 16GB GPU.

To collect documents, CiteSeerX crawls researchers' homepages, URLs from the Microsoft Academic Graph and Google Scholar, and maintains whitelists and blacklists for crawling. Our framework is integrated in CiteSeerX to continuously augment and maintain the URLs collection used for crawling (in order to preserve network bandwidth and hardware). CiteSeerX also directly incorporates PDFs from PubMed, arXiv, and digital repositories in a diverse spectrum of disciplines such as mathematics, physics, and medical science. These crawled documents are passed to multiple AI modules such as document classification, document de-duplication and citation graph, metadata extraction, header extraction, citation extraction, etc. The ingestion module writes all metadata into the database. The PDF documents are renamed under a document ID (csxDOI) and saved to the production repository with XML metadata files. The index data are also updated. During the application development, we have learned that identifying homepages "in the wild" is very challenging since

they have very diverse structures and content. Moreover, based on the human annotation task, identifying author homepages is difficult sometimes even for human annotators.

3.7 Maintenance of Researchers' Homepages Discovery Framework in CiteSeerX

The researcher homepage classifier is developed and maintained by one graduate student, whereas the web-crawler component in CiteSeerX is developed and maintained by several graduate students. The homepages finding project received partial financial support from the National Science Foundation aimed at a sustainable CiteSeerX. While collecting the documents from homepages (seed URLs), multiple types of documents can be found on homepages such as CVs, slides, syllabus, homeworks, etc. which should not be included in CiteSeerX. Thus we found that a classifier that distinguishes research articles from other types of documents, as described in[100], should be used on the crawled documents. During maintenance, as new homepages emerge and also existing authors may change affiliations or the homepage may get outdated (4XX error), periodically we need to automatically update the list of homepages as well as remove the outdated homepages without the human effort. The maintenance work includes, but is not limited to fixing bugs, updating the list of URLs including researcher homepages, periodically checking the system health, and running the web-crawlers. CiteSeerX data is updated regularly. The crawling rate varies from 50,000 to 100,000 PDF documents per day. Of the crawled documents, about 40% are eventually identified as being academic and ingested into the database.

3.8 Summary and Future Directions

We proposed a framework for automatically acquiring research papers from the Web. We showed the experiments illustrating the state-of-the-art performance for two major modules of our framework: a homepage identifier and a paper classifier. Through an experiment using

a large collection of $\approx 76,000$ queries (titles + authors names), our framework was able to automatically acquire an overall collection of $\approx 267,000$ unique research papers and was able to recover 78% of the original searched titles, i.e., $\approx 34,000$ papers from the $43,496$ original searched titles. We also showed that our approach is not meant to replace existing crawling strategies, but can be used in conjunction, to enhance the content of digital libraries.

Moreover, we proposed another novel search-then-classify approach to discover researcher homepages using author names and paper titles as queries in order to augment and maintain the URL lists for document crawling in CiteSeerX. To our knowledge, we are the first to interleave Web search and deep learning for researcher homepage identification to build an efficient author homepage acquisition approach. This is a useful component in CiteSeerX, which crawls researchers' homepages to collect research papers for inclusion in the library. Moreover, we show that self-training can be very useful to train deep learning based researcher homepage classifiers using small amount of labeled data along with unlabeled data. Since data annotation is very expensive, we show that human effort can be reduced through self-training, which could be useful when deploying this into another system in future. Our results showcase the potential of our approach. More interestingly, we discovered $12,199$ researcher homepages using $10,000$ paper title queries. This shows the capability of research paper titles for finding researcher homepages. We show the integration of our framework in CiteSeerX for collecting URLs for crawling scientific documents. The new datasets that we created are made available online to foster research in this area.

In the future, it would be interesting to apply our frameworks to other domains, and study the integration of topic classification.

Chapter 4

Document Classification in Web Archiving Collections

The Web archived data usually contains high-quality documents that are very useful for creating specialized collections of documents. To create such collections, there is a substantial need for automatic approaches that can distinguish the documents of interest for a collection out of the large collections (of millions in size) from Web Archiving institutions. However, the patterns of the documents of interest can differ substantially from one document to another, which makes the automatic classification task very challenging. In this chapter, we explore different learning models and feature representations to determine the best performing ones for identifying the documents of interest from the web archived data. Specifically, we study both machine learning and deep learning models and "bag of words" (BoW) features extracted from the entire document or from specific portions of the document, as well as structural features that capture the structure of documents. Moreover, we explore dynamic fusion models to find, on the fly, the model or combination of models that performs best on a variety of document types. We focus our evaluation on three datasets that we created from three different Web archives. Our experimental results show that the approach that fuses different models outperforms individual models and other ensemble methods on all three datasets.

4.1 Introduction

A growing number of research libraries, museums, and archives around the world are embracing web archiving as a mechanism to collect born-digital material made available via the web. Between the membership of the International Internet Preservation Consortium, which has 55 member institutions[142], and the Internet Archive's Archive-It web archiving platform with its 529 collecting organizations[143], there are hundreds of institutions currently engaged in building collections with web archiving tools. The amount of data that these web archiving initiatives generate is typically at levels that dwarf traditional digital library collections. As an example, in a recent impromptu analysis, Jefferson Bailey of the Internet Archive noted that there were 1.6 Billion PDF files in the Global Wayback Machine[25]. If just 1% of these PDFs are of interest for collection organizations, that would result in a collection larger than the 15 million volumes in HathiTrust[26].

While the number of Web Archiving institutions increases, the technologies needed to provide access to these large collections have not improved significantly over the years. At this time, the standard way of accessing web archives is with known URL lookup using tools like the OpenWayback[1] or pywb[2]. The use of full-text search has increased in many web archives around the world, but often provides an experience that is challenging for users because of the vast amount of content and the limitations of strictly text-based searches for these large heterogeneous collections of content. Another avenue of access to web archived data that is of interest to Web Archiving institutions is the ability to extract high-quality, content-rich publications from the web archives in order to add them to their existing collections.

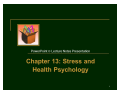
Our research is aimed at understanding how well machine learning and deep learning models can be employed to provide assistance to collection maintainers who are seeking to classify the PDF documents from their web archives into being within scope for a given collection or collection policy or out of scope. By identifying and extracting these documents, institutions will improve their ability to provide meaningful access to collections of materials harvested from the web that are complementary, but oftentimes more desirable than tradi-

[1] https://github.com/iipc/openwayback
[2] https://github.com/webrecorder/pywb

tional web archives. At the same time, building specialized collections from web archives shows usage of web archives beyond just replaying the web from the past. Our research focus is on three different use cases that have been identified for the reuse of web archives and include populating an institutional repository from a web archive of a university domain, the identification of state publications from a web archive of a state government, and the extraction of technical reports from a large federal agency. These three use cases were chosen because they have broad applicability and cover different Web archive domains.

Precisely, in this chapter, we explore and contrast different learning models and types of features to determine the best performing ones for identifying the documents of interest that are in-scope of a given collection. Our study includes both an exploration of traditional machine learning models in conjunction with either a "bag of words" representation of text or structural features that capture the characteristics of documents, as well as an exploration of Convolutional Neural Networks (CNN). The "bag of words" (BoW) or *tf-idf* representation is commonly used for text classification problems[144;145]. Structural features designed based on the structure of a document have been successfully used for document type classification[100]. Moreover, for text classification tasks, Convolutional Neural Networks are also extensively used in conjunction with word embeddings and achieve remarkable results[128;146;147].

In Web Archiving collections, usually, the documents are very diverse, with different types of documents having a different textual structure and covering different topics. As an example, consider a scholarly works repository, which contains publications that are typical for an institutional repository such as research articles, white papers, slide decks from presentations, and other scholarly publications. Documents not considered as part of the scholarly works repository include curriculum vitae, resumes, publications lists, and student manuals. The beginning and the end portions of a document might contain useful and sufficient information (either structural or topical) for deciding if a document is in scope of a collection or not. For example, research articles usually contain the abstract and the introduction in the beginning of the document, with the conclusion, acknowledgements, and references occurring towards the end of the document. Being on the proper subject (or in scope of a collection) can often also be inferred from the beginning and the end portions

Single classifier is correct			
Document	Class Label	Description	Prob.
	+ve	Slides	BoW: 0.15 **Str: 0.74** CNN: 0.16 SEM: 0.35
		(a)	
	-ve	CV	BoW: 0.55 **Str: 0.40** CNN: 0.87 SEM: 0.61
		(b)	

Two classifiers are correct			
Document	Class Label	Description	Prob.
	+ve	Research article	BoW: 0.40 **Str: 0.85** **CNN: 0.63** **SEM: 0.63**
		(c)	
	-ve	Publication list	**BoW: 0.45** **Str: 0.11** CNN: 0.97 SEM: 0.51
		(d)	

Figure 4.1: *Example documents from a Web Archiving collection and classifiers' confidences.*

of a document. To this end, we consider the task of finding documents being in-scope of a collection as a binary classification task. In our work, we experiment with bag of words ("BoW") by using text from the entire document as well as by focusing only on specific portions of the document (i.e., the beginning and the end part of the document). Although structural features were originally designed for document type classification, we used these features for our binary classification task. We also experiment with a CNN classifier that exploits pre-trained word embeddings.

Moreover, we conjecture that simply using any of the above individual classifiers or combining them with static decision-level fusion (e.g., aggregating all three BoW, Str, and CNN classifiers' confidences) may not always help in finding relevant documents to a particular repository from Web Archiving collections that contain a wide variety of documents. Figure 4.1 illustrates this phenomenon using a few examples sampled from one of our datasets (a scholarly works repository). The figure contains the predicted probabilities for textual content based BoW and CNN classifiers, structural features classifier (Str), and static ensemble model (SEM) that averages the probability of each individual classifier (BoW, Str, and CNN, in our case). For the document (a) (slides) in Figure 4.1, only Str classifier correctly predicts the class label with a probability of 0.74. However, BoW, CNN and SEM miss-classified the document with very low probability of 0.15, 0.16, and 0.35, respectively. Similarly for document (b) (CV), only Str classifier correctly predicts the document as out of

collection, while all others (BoW, CNN, and SEM) mistakenly classify the CV as being part of the repository. Interestingly, for the research article (document) (c), only BoW classifier miss-labeled the document with 0.40 probability, whereas Str, CNN, and SEM correctly classified the document. Similar observations can be seen for the document (d) (a publications list). To this end, we further explore dynamic decision-level classifier selection or fusion to identify on the fly the best suited classifier or combination of classifiers that can on a variety of document types.

Our contributions are as follows:

- We built three datasets from three different web archives collected by the UNT libraries, each covering different domains: UNT.edu, Texas.gov, and USDA.gov. Each dataset contains the PDF document along with the label indicating whether a document is in scope of a collection or not. We will make these datasets available to further research in this area.

- We show that BoW classifiers that use only some portion of the documents outperform BoW classifiers that use full text from the entire content of a document, the structural features based classifiers, and the CNN classifier. We also show that feature selection using information gain improves the performance of the BoW classifiers and structural features based classifiers, and present a discussion on the most informative features for each collection.

- We propose a dynamic classifier selection for document classification (DCSDC) to dynamically select an appropriate classifier to predict the probability of a target document as being in scope of a collection or not. To dynamically select the classifiers, we consider textual similarity along with the structural aspects of the documents. We show that DCSDC outperforms all the individual feature set models (base classifiers) and other strong baselines.

- We propose a dynamic decision-level fusion for document classification (DDFC) that derives competence features from neighborhood documents and learns a classifier to

assign a competence score for each base classifier (BoW, Str, and CNN) in order to fuse them and to predict the probability of a target document as being in scope of a collection or not. To derive the competence features, we consider textual similarity along with the structural aspects of the documents. We show that DDFC outperforms all the individual feature set models (base classifiers) and other strong baselines including DCSDC.

4.2 Related Work

Web Archiving. Web archiving as a method for collecting content has been conducted by libraries and archives since the mid-1990's. The most known web archiving is operated by the Internet Archive who began harvesting content in 1996. Other institutions throughout the world have also been involved in archiving the web based on their local collection context whether it is based on a subject or as part of a national collecting mandate such as with national libraries across the world. While the initial focus of these collections was to preserve the web as it exists in time, there were subsequent possibilities to leverage web archives to improve access to resources that have been collected, for example, after the collaborative harvest of the federal government domain by institutions around the United States for the 2008 End of Term Web Archive whose goal was to document the transitions from the Bush administration to the Obama administration. After the successful collection of over 16TB of web content, Phillips and Murray[148] analyzed the 4.5M unique PDFs found in the collection to better understand their makeup. Jacobs[149] articulated the value and importance of web-published documents from the federal government that are often found in web archives. Nwala et al.[150] studied bootstrapping of the web archive collections from the social media and showed that sources such as Reddit, Twitter, and Wikipedia can produce collections that are similar to expert generated collections (i.e., Archive-It collections). Alam et al.[151] proposed an approach to index raster images of dictionary pages and built a Web application that supports word indexes in various languages with multiple dictionaries. Alam et al.[152] used CDX summarization for web archive profiling, whereas AlNoamany et al.[153] proposed the

Dark and Stormy Archive (DSA) framework for summarizing the holdings of these collections and arranging them into a chronological order. Aturban et al.[154] proposed two approaches to to establish and check fixity of archived resources. More recently, the Library of Congress[155] analyzed its web archiving holdings and identified 42,188,995 unique PDF documents in its holdings. These initiatives show interest in analyzing the PDF documents from the web as being of interest to digital libraries. Thus, there is a need however for effective and efficient tools and techniques to help filter or sort the desirable PDF content from the less desirable content based on existing collections or collection development policies. We specifically address this with our research agenda and formulate the problem of classifying the PDF documents from the web archive collection into being of scope for a given collection or being out of scope. We use both traditional machine learning and deep learning models. Below we discuss related works on both of these lines of research.

Ensemble models and dynamic fusion. Ensemble models have been used previously in many systems[156–158]. Bagging is an ensemble technique that builds a set of diverse classifiers, each trained on a random sample of the training data to improve the final (aggregated) classifiers' confidence[156;158]. Since the classifiers in an ensemble may learn very different patterns, dynamic ensembles that extend bagging have also been proposed[159–161]. In dynamic ensembles, a pool of classifiers are trained on a single feature type (e.g., bag-of-words), each using a different subset of examples or features, within the bagging technique[156;158] and the competence of the base classifiers is determined dynamically. Our work extends these approaches to exploit various feature types or classifiers to capture different aspects of documents (structure and topicality) to perform classifiers' fusion for document categorization (rather than one single feature type).

Ensemble classifiers have also been used in a multi-modal setting[162;163], in which different modalities are coupled, e.g., images and text for image retrieval[164] and image classification[162]. Zahavy et al.[165] urged the development of optimal unification methods to combine different classifiers trained on different modalities. Co-training approaches[166] use multiple views of the data to "guide" different classifiers in the learning process. However, co-training methods are semi-supervised and assume that all views are "sufficient" for learning. In contrast with

the above approaches, we aim to capture different aspects of documents (structure and topicality), with each aspect having a different competence power, and perform dynamic selection of classifiers for classifying the textual documents from a Web Archiving collection into being in scope for the collection or not.

Traditional Text Classification. Text classification is a well-studied problem. The *BoW* (*binary*, *tf*, or *tf-idf*) representations are commonly used as input to machine learning classifiers, e.g., Support Vector Machine[167] and Naïve Bayes Multinomial[168] for text classification. Feature selection is often applied to these representations to remove irrelevant or redundant features[169;170]. In the context of digital libraries, the classes for text classification are often document topics, e.g., papers classified as belonging to "machine learning" or "information retrieval"[7]. Structural features that capture the structural characteristics of documents are also used for the classification of documents in digital libraries[100]. Comprehensive reviews of the feature representations, methods, and results on various text classification problems are provided by Sebastiani[145] and Manning[171]. Craven and Cumlien[172] classified bio-medical articles using the Naive Bayes classifier. Kodakateri Pudhiyaveetil et al.[173] used the k-NN classifier to classify computer science papers into 268 different categories based on the ACM classification tree. Other authors[174;175] experimented with different classification methods such as unigram, bigram, and Sentence2Vec[176] to identify the best classification method for classifying academic papers using the entire content of the scholarly documents.

Deep Learning. Deep learning models have achieved remarkable results in many NLP and text classification problems[39;136;146;177–179]. Most of the works for text classification with deep learning methods have involved word embeddings. Among different deep learning architectures, convolutional neural network (CNN), recurrent neural network (RNN), and their variations are the most popular architectures for text applications. Kalchbrenner et al.[179] proposed a deep learning architecture with multiple convolution layers that uses word embeddings initialized with random vectors. Zhang et al.[180] used encoded characters ("one-hot" encoding) as an input to the deep learning architecture with multiple convolution layers. They proposed a 9-layer deep network with 6 convolutional layers and 3 fully-connected layers. Kim[181] used a single layer of CNN after extracting word embeddings for tokens in

the input sequence. The author experimented with several variants of word embeddings, i.e., randomly initialized word vectors later tuned for a specific task, fixed pre-trained vectors, pre-trained vectors later tuned for a specific task, and a combination of the two sets of word vectors. Yin et al.[182] used the combination of diverse versions of pre-trained word embeddings followed by a CNN and a fully connected layer for the sentence classification problem.

4.3 Datasets

For this research, we constructed datasets from three web archives collected by the UNT Libraries. For each of the datasets we extracted all PDF documents within each of the web archives. Next, we randomly sampled 2,000 PDFs from each collection that we used as the basis for our labeled datasets. Each of the three sets of 2,000 PDF documents were then labeled in scope and out of scope by subject matter experts who are responsible for collecting publications from the web for their collections. Each dataset includes PDF files along with their labels (in scope/out of scope or relevant/irrelevant). Further description of the datasets is provided below.

4.3.1 UNT.edu Dataset

The first dataset was created from the UNT Scholarly Works web archive of the unt.edu domain. This archive was created in May 2017 as part of a bi-yearly crawl of the unt.edu domain by the UNT Libraries for the University Archives. A total of 92,327 PDFs that returned an HTTP response of 200 were present in the archive. A total of 3,141,886 URIs were present in the entire web archive with PDF content making up just 3% of the total number of URIs.

A set of 2,000 PDFs were randomly sampled from the full set of PDF documents and were given to two annotators for labeling. These annotators were: one subject matter expert who was responsible for the maintenance of the UNT Scholarly Works Repository and one

graduate student with background in Library and Information Systems. They proceeded to label each of the PDF documents as to whether a document would be of interest to the institutional repository or if the document would not be of interest. The labeling of the PDF files resulted in 445 documents (22%) identified as being of interest for the repository and 1,555 not being of interest. In case of disagreement between annotators, a final decision was made by the researchers of this paper after a discussion with the annotators.

4.3.2 Texas.gov Dataset

The next dataset was created from a web archive of websites that constitute the State of Texas web presence. The data was crawled from 2002 until 2011 and was housed as a collection in the UNT Digital Library. A total of 1,752,366 PDF documents that returned an HTTP response of 200 were present in the archive. A total of 26,305,347 URIs were present in the entire web archive with PDF content making up 6.7% of the total number of URIs.

As with the first dataset, a random sample of 2,000 PDF documents was given to two annotators for labeling: a subject matter expert from the UNT Libraries and a graduate student (as before). In this case, items were identified as either being in scope for a collection called the "Texas State Publications Collection" at the UNT Libraries, or out of scope. This collection contains a wide range of publications from state agencies. The labeling of the PDF files resulted in 136 documents (7%) identified as being of interest for the repository and 1,864 not being of interest. Again, in case of disagreement between annotators, a final decision was made by the researchers of this paper after a discussion with the annotators.

4.3.3 USDA.gov Dataset

The last dataset created for this study came from the End of Term (EOT) 2008 web archive. This web archive was created as a collaborative project between a number of institutions at the transition between the second term of George W. Bush and the first term of Barack Obama. The entire EOT web archive contains 160,212,141 URIs. For this dataset we selected

the United States Department of Agriculture (USDA) and its primary domain of usda.gov. This usda.gov subset of the EOT archive contains 2,892,923 URIs with 282,203 (9.6%) of those being PDF files that returned an HTTP 200 response code.

Similar to the previous datasets, a random sample of 2,000 PDF documents was given to two annotators for labeling: a subject matter expert who has worked as an engineering librarian for a large portion of their career and a graduate student (as before). The subject matter expert has also been involved with the Technical Report Archive and Image Library (TRAIL) that was collecting, cataloging, and digitizing technical reports published by, and for, the federal government throughout the 20th century. The annotators labeled each of the PDF files as either being of interest for inclusion in a collection of Technical Reports or not being of interest to that same collection. The final labeled dataset has 234 documents (12%) marked as potential technical reports and 1,766 documents identified as not being technical reports. The disagreements between the annotators were finally adjudicated by one of the researchers of this paper.

The three datasets represent a wide variety of publications that would be considered for inclusion into their target collections. Of these three, the Texas.gov content is the most diverse as the publications range from strategic plans and financial audit reports (which are many pages in lengths) to pamphlets and posters (which are generally very short). The UNT.edu dataset contains publications that are typical for an institutional repository such as research articles, white papers, slide decks from presentations, and other scholarly publications. The publications from the UDSA.gov dataset are similarly scoped as the UNT.edu content, but they also contain a wider variety of content that might be identified as a "technical report." A goal in the creation of the datasets used in this research was to have a true representative sample of the types of content that are held in collections of this kind.

4.4 Base Classifiers

In this section, we discuss different types of features that are used in conjunction with traditional machine learning classifiers for finding documents of interests, and the CNN model that does not require any feature engineering. These models form our base classifiers.

4.4.1 Bag of Words (BoWs)

"Bag of words" (BoW) is a simple fixed-length vector representation of any variable length text based on the occurrence of words within the text, with the information about the positions of different words in a document being discarded. First, a vocabulary from the words in the training documents is generated. Then, each document is represented as a vector based on the words in the vocabulary. The values in the vector representation are usually calculated as normalized term frequency (*tf*) or term frequency - inverse document frequency (*tf-idf*) of the corresponding word calculated based on the given document/text.

We experiment with BoW extracted from the full text of the documents as well as from only some portions of documents. Our intuition behind using only some portion of the documents is that many types of documents contain discriminative words at the beginning and/or at the end. For selecting these portions of documents, we consider first-X words from each document, and first-X words combined with last-X words from each document before any type of preprocessing was performed. We experimented with values of $X \in \{100, 300, 500, 700, 1000, 2000\}$. For documents with less than $2 \cdot X$ words, we considered the entire document without repeating any parts/words from the document.

Moreover, for BoW encoded from the full text of documents, we also compared the performance of the top-N selected features, using the information gain (IG) feature selection method, where $N \in \{300, 500, 1000, 2000, 3000\}$, with the performance of all features.

4.4.2 Structural Features

Structural features (Str) are designed to incorporate aspects specific to documents' structure and are shown to be highly indicative of the classification of academic documents into their document types such as Books, Slides, Theses, Papers, CVs, and Others[100].

These features can be grouped into four categories: file specific features, text specific features, section specific features, and containment features. Each of these feature categories are described below.

File specific features include the characteristics of a document such as the number of pages and the file size in kilobytes.

Text specific features include specifics of the text of a document: the length in characters; the number of words; the number of lines; the average number of words and lines per page; the average number of words per line; the count of reference mentions; the percentage of reference mentions, spaces, uppercase letters, symbols; the ratio of length of shortest to the longest line; the number of lines that start with uppercase letters; the number of lines starting with non-alphanumeric letters; the number of words that appear before the reference section.

Section specific features include section names and their position within a document. These features are boolean features indicating the appearance of "abstract", "introduction", "conclusion", "acknowledgements", "references" and "chapter," respectively, as well as numeric features indicating position for each of these sections. These features also include two binary features indicating the appearance of "acknowledgment" before and after "introduction."

Containment features include containment of specific words or phrases in a document. These features include binary features indicating the appearance of "this paper," "this book," "this thesis," "this chapter," "this document," "this section," "research interests," "research experience," "education," and "publications," respectively. These features also include three numeric features indicating the position of "this paper," "this book," and "this thesis" in a

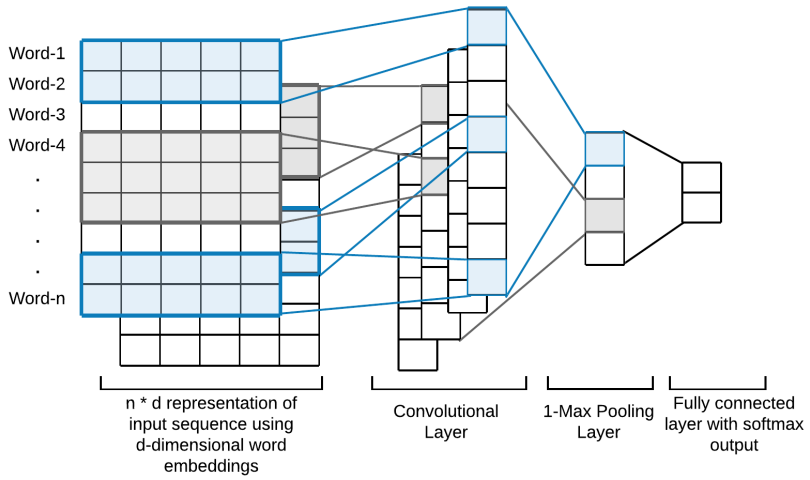


Figure 4.2: *CNN architecture for classification.*

document.

Similar to BoW, for the structural features (which are 43 in total), we also compared the performance of the top-N selected features, ranked using the information gain (IG) feature selection method, where $N \in \{10, 20, 30\}$, with the performance of all 43 features.

4.4.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNN or ConvNets)[135] are a special kind of neural networks to process grid-like structured data, e.g., image data. CNNs are associated with the idea of a "moving filter." A convolution consists of a filter or a kernel, that is applied in a sliding window fashion to extract features from the input. This filter is shifted after each operation over the input by an amount called strides. The convolution layer consists of multiple filters of different region sizes that generate multiple feature maps for different region sizes. Pooling is usually used after the convolution layer to modify the output or reduce the dimensionality. The common practice is to extract the most important feature within each feature map[128;136], called 1-max pooling. Max pooling is applied over each feature map and the maximum values from each filter are selected. Maximum values from each feature map are then concatenated and used as input to a fully connected layer for the classification task. Generally, pooling

helps to make the representation become approximately invariant to small changes in the input. The CNN architecture that we used in our experiments is shown in Figure 4.2 and is similar to the CNN architecture developed by Kim[128].

For CNN, we experimented with using the text from specific portions of the document. While selecting the portions of documents, as before, we considered first-X words and first-X words combined with last-X words from each document before any preprocessing was performed (where $X \in \{100, 300, 500, 700, 1000\}$). For the documents with less than $2 \cdot X$ words, we considered the whole document without repeating any part/words from the document.

4.5 Proposed Model: Dynamic Classifier Selection

In contrast with the approaches that use a single model to identify relevant documents to a given collection, we propose an approach called "Dynamic Classifier Selection for Document Classification" (or DCSDC), that dynamically selects an appropriate classifier to identify if a given document is relevant to a collection by dynamically capturing different aspects of the document. The intuition behind using this approach is that there is a high variability in the type of the documents in each dataset.

The proposed approach consists of a three-step process for classifying a given document:

- **Step-1: Find its neighborhood documents.** We identify the neighborhood documents of the target document by considering textual similarity along with their structural aspects using K-Nearest Neighbors algorithm ($K \in \{5, 10, 15, 20, 50\}$). In the first step, we consider the documents in the **Dev** set that fall under the range of $X \pm L$ pages (X is the number of pages of the target document and L is a page range limit; we consider $L = 3$). After that, we calculate the textual similarity between the target document and those documents from **Dev** that are within $X \pm L$ page range from the target document. In order to calculate the textual similarity between the documents, we use

two different methods by considering top N most frequent words ($N \in \{25, 50, 100\}$): (a) tf-idf based cosine similarity, and (b) Word centroid based cosine similarity, where the centroid is calculated by a weighted average word vectors. We consider pre-trained word embeddings trained on Google News for the word vectors.

- **Step-2: Find the most competent classifier.** Here we find the set of features and classifiers that perform best on neighborhood documents. The goal is to find the most competent classifier for a particular type of document. We apply each individual or the combination of different feature sets to neighborhood documents and find which classifier has the highest success rate for labeling them correctly.

- **Step-3: Use the most competent classifier on the test example.** The classifier with the highest success rate based on step-2 is used to classify the given test document. In case of multiple classifiers with the highest success rate, we selected the majority vote.

4.6 Proposed Model: Dynamic Decision Level Fusion

In contrast with the approaches that use a single model to identify relevant documents for a given collection, we propose an approach called "Dynamic Decision level Fusion for Document Classification" (or DDFC), that effectively blends different classifiers to identify if a document is relevant to a collection by dynamically capturing different aspects of the document. The intuition behind using this approach is that there is a high variability in the type of the documents in Web archiving collections and, for a given document, only a small subset of base classifiers might be useful for its classification. For example, in Figure 1(a) only the Str classifier is competent, and hence, useful. Our approach specifically learns base classifiers' competences and dynamically selects one or more base classifiers at decision level, based on their competence (or expertise) for a given document.

DDFC makes use of two datasets, indicated as $\mathcal{D}^{Tr}$ and $\mathcal{D}^{Dev}$ containing both positive

Notation	Description
$\mathcal{D}^{Tr}$	$= \{(T_1, L_1), \cdots, (T_m, L_m)\}$ a dataset of m labeled documents for base classifier training.
$\mathcal{D}^{Dev}$	$= \{(V_1, L_1), \cdots, (V_n, L_n)\}$ a dataset of n labeled documents for competence estimation.
T	A target document.
$\mathcal{B}$	$= \{B^b, B^s, B^c\}$ trained on $\mathcal{D}^{Tr}$ using BoW, Str, and CNN, respectively.
N^T	The neighborhood documents of T by considering textual similarity along with their structural aspects.
k_N	The size of N^T, where $1 \leq k_N < n$.
C	$= \{C^b, C^s, C^c\}$ a set of "competence" classifiers corresponding to the base classifiers from $\mathcal{B}$ (e.g., C^b for B^b).
Φ	$= \{\phi^b, \phi^s, \phi^c\}$ a set of "competence" feature vectors to train the "competence" classifiers.

Table 4.1: *Notations.*

and negative examples. We use $\mathcal{D}^{Tr}$ to train our base classifiers. Particularly, we train three base classifiers $\mathcal{B} = \{B^b, B^s, B^c\}$ on $\mathcal{D}^{Tr}$, BoW classifier, Str classifier, and CNN classifier, respectively. The competence (or expertise) of these base classifiers are learned on $\mathcal{D}^{Dev}$ (Section 5.2.2). For a target document T, we first identify its neighborhood documents from $\mathcal{D}^{Dev}$ by considering textual similarity along with their structural aspects using the K-Nearest Neighbors algorithm (Section 4.6.1). Second, we estimate the competence of each base classifier (or their expertise in classifying T) by using the neighborhood documents found in the first step (Section 4.6.2). Last, we use the competence classifiers and their competence scores to dynamically select base classifiers to predict the class label for T (Section 4.6.3). Figure 4.3 shows an illustration of our approach at test time. We explain the stages of DDFC in detail below. The notation used is provided in Table 4.1.

4.6.1 Step-1: Finding Neighborhood Documents

For a document T, we identify the neighborhood documents N^T from $\mathcal{D}^{Dev}$ by considering textual similarity along with their structural aspects using the K-Nearest Neighbors algorithm (K or $k_N \in \{5, 10, 15, 20, 50\}$). We first consider the documents which fall under the range of $X \pm L$ pages (X is number of pages of a given document and L is a page range limit, we consider $L = 3$). After that, we calculate the similarity between the document T and each of the documents in $\mathcal{D}^{Dev}$ that fall within the required page range, and select the top k_N

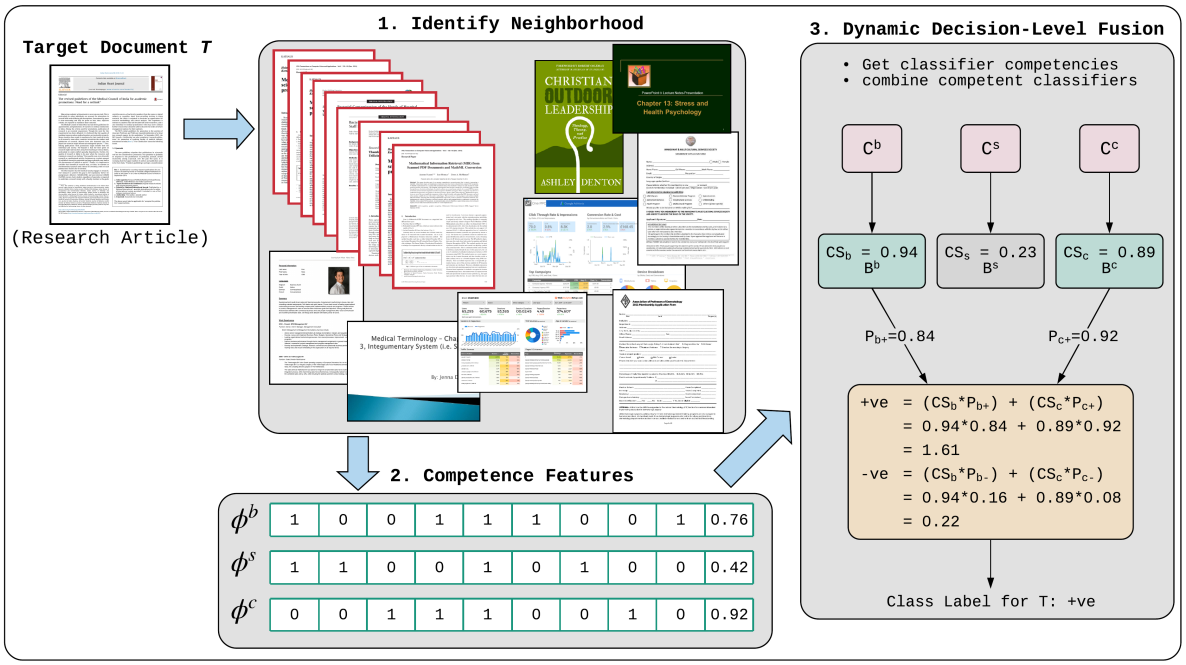


Figure 4.3: *Illustration of our approach using an example.*

most similar documents to T. In order to calculate the similarity between the documents, we use two different methods by considering up to N most frequent words ($N \in \{25, 50, 100\}$): (a) tf-idf based cosine similarity, and (b) Word centroid based cosine similarity, where the word centroid is calculated by a weighted average word vectors. We consider pre-trained word embeddings trained on Google News for the word vectors.

4.6.2 Step-2: Competence Estimation

Here, we discuss the competence estimation of each base classifier.

Extracting "Competence" Features.

We use two sets of features to learn each competence classifier on $\mathcal{D}^{Dev}$. The first set of features ϕ_1 for a document T is obtained using the neighborhood N^T and the second feature ϕ_2 utilizes the probability of T being in scope of the collection, $P(+|T, B_i)$, as predicted by the base classifiers, B_i. We concatenate both set of features to generate the "competence"

Algorithm 1 Learning Competence Classifiers

Input: A dataset $\mathcal{D}^{Dev} = \{(V_1, L_1), \cdots, (V_n, L_n)\}$ of labeled documents; neighborhood size k_N; pre-trained word embeddings $\mathcal{E}$; a set of base classifiers $\mathcal{B} = \{B^b, B^s, B^c\}$.
Output: A set of "competence" classifiers $\mathcal{C} = \{C^b, C^s, C^c\}$.
$\mathcal{F} = \{F^b, F^s, F^c\} \leftarrow \varnothing$; // Examples for training competence classifiers, initially empty.
$\mathcal{C} \leftarrow \{\}$; // A set of competence classifiers, initially empty.
for all $V_j \in \mathcal{D}^{Dev}$ **do**
    $N^{V_j} \leftarrow IdentifyNeighborhood(k_N, V_j, \mathcal{E}, \mathcal{D}^{Dev})$; // k_N textually and/or structurally similar documents of V_j.
    for all $B_i \in \mathcal{B}$ **do** // Iterate through the set of base classifiers.
        $\phi_{i,j} \leftarrow CompetenceFeatures(V_j, N^{V_j}, B_i)$;
        if $Predict(B_i, V_j) = L_j$ **then** // predicted correctly.
            $L_{ij} \leftarrow 1$; // B_i is competent for V_j.
        else
            $L_{ij} \leftarrow 0$; // B_i is not competent for V_j.
        end if
        $F^i \leftarrow F^i \cup \{(\phi_{i,j}, L_{ij})\}$
    end for
end for
for all $F^i \in \mathcal{F}$ **do** // Train competence classifiers.
    $C_i \leftarrow TrainCompetenceClassifier(D^i)$;
    $\mathcal{C} \leftarrow \mathcal{C} \cup C_i$
end for
return $\mathcal{C}$;

feature vector of length $k_N + 1$. We generate the "competence" feature vector corresponding to each base classifier.

- ϕ_1: For T, a vector of length k_N capturing the correctness of a base classifier calculated using the neighborhood documents N^T. An entry j in ϕ_1 is 1 if a base classifier $B_i \in \mathcal{B}$ accurately predicts the document j being in scope or not $X_j \in N^T$, and is 0 otherwise, where $j = 1, \cdots, k_N$. For T in Figure 4.3, $\phi_1 = \{1, 0, 0, 1, 1, 1, 0, 0, 1\}$, obtained by B^b.
- ϕ_2: For T, we use the class probability predicted by a given base classifier $P(+|T, B_i)$, where $B_i \in \mathcal{B}$. In Figure 4.3, for the target document T, $\phi_2 = 0.76$, obtained using B^b.

Next, we discuss how we learn a competence classifier for each base classifier. Note that the competence classifiers are learned only once corresponding to each base classifier using documents from $\mathcal{D}^{Dev}$.

"Competence" Learning.

For calculating the competence score for each base classifier we train a "competence" classifier for each base classifier on $\mathcal{D}^{Dev}$. A competence classifier predicts the competence score for the corresponding base classifier. Algorithm 1 explains the steps for "competence" learning. To train the "competence" classifiers, each document from $\mathcal{D}^{Dev}$ is considered as a target document one by one. For each document, neighborhood documents from $\mathcal{D}^{Dev}$ are collected (Alg. 1, line 6). Then for each base classifier, competence features are extracted as explained earlier in step-2, and we consider label as 1 if the base classifier B_i has predicted the document correctly Alg. 1, lines 8-14). At the end, "competence" classifiers (C^b, C^s, and C^c) corresponding to each base classifier (B^b, B^s, and B^c) are trained (Alg. 1, lines 17-20).

4.6.3 Step-3: Dynamic Decision-Level Fusion

In this step, For a target document T, we dynamically combine a subset of base classifiers. If not all the base classifiers agree (Algo. 2, line 6), then we first find the neighborhood documents for T (Algo. 2, line 7). Then, for each base classifier the competence features are extracted as we described in step-2 and given to the corresponding "competence" classifiers (Algo. 2, line 9,10). A "competence" score CS_i is output as a probability of base classifier B_i being competent. The base classifiers with the competence score > 0.5 are selected to classify T. To predict T, we use the weighted sum of the probability of individual class label assigned by each selected base classifier with CS_i as the weight for B_i, and the class label with the highest probability is considered as the predicted label (Algo. 2, line 16).

Illustration of the proposed method.

Figure 4.3 shows the illustration of our approach using an anecdotal example. We consider a research article as a target document T. First, we identify $k_N = 9$ neighborhood documents for T, which are highlighted in red. From these neighborhood documents and the

Algorithm 2 Dynamic Decision Level Fusion

Input: A target document T; $\mathcal{D}^{Dev} = \{(V_1, L_1), \cdots, (V_n, L_n)\}$ a dataset of labeled documents; neighborhood size k_N; pre-trained word embeddings $\mathcal{E}$; a set of base classifiers $\mathcal{B} = \{B^b, B^s, B^c\}$; and a set of competence classifiers $\mathcal{C} = \{C^b, C^s, C^c\}$.
Output: Class label L_T.
$B' \leftarrow \{\}$; // the subset of most competent base classifiers.
$CS \leftarrow \{\}$; // the set of competence scores.
$BA_T \leftarrow Agreement(\mathcal{B}, T)$; // Base classifiers' agreement on T's label.
if $BA_T \leq |\mathcal{B}|$ **then**
    $N^T \leftarrow IdentifyNeighborhood(k_N, T, \mathcal{E}, \mathcal{D}^{Dev})$; // k_N textually and/or structurally similar documents of T.
    for all $B_i \in \mathcal{B}$ & $C_i \in \mathcal{C}$ **do** // Iterate through the set of base and competence classifiers.
        $\phi_i \leftarrow CompetenceFeatures(T, N^T, B_i)$;
        $CS_i \leftarrow PredictCompetence(\phi_i, C_i)$; // Predict competence score for base classifier B_i.
        if $CS_i > 0.5$ **then** // If the predicted competence score is greater than 0.5 then the base classifier B_i is competent.
            $B' \leftarrow B' \cup \{B_i\}$
            $CS \leftarrow CS \cup \{CS_i\}$
        end if
    end for
    $L_T = WeightedScoreLabel(T, B', CS)$ // weighted sum of individual class probabilities are calculated and the highest score is taken.
end if
return L_T;

base classifiers' predicted probability for T, we generate the "competence" feature vectors corresponding to each base classifier (ϕ^b, ϕ^s, and ϕ^c). Then, the "competence" classifiers (C^b, C^s, and C^c) are used on the corresponding "competence" feature vectors to find the competence score for each base classifier (CS_b, CS_s, and CS_c). At last, the base classifiers with the competence score > 0.5 are used to predict the class label for T by doing weighted sum of the individual class probabilities for all the selected base classifiers with CS_i as the weight for B_i, and selecting the class label with the highest probabilty.

4.7 Experiments and Results

In this section, we first discuss the baselines and then the experimental setup for our document classification task. Next, we present the preliminary set of experiments to fix base classifier parameters and to find which portion of a document to use on different classifiers. We then present an analysis to highlight the potential of the proposed algorithm. At last, we compare our proposed approach with the individual base classifiers and strong baselines.

4.7.1 Baselines

The baselines used for comparison are described below.

1. KNN: For a target document T, we calculate its K-nearest neighbors from $\mathcal{D}^{Tr}$ and then select the majority class label from the neighbors set. We experiment with $K \in \{1, 5, 9, 15\}$ and select the best value of K using the $\mathcal{D}^{Dev}$ set.

The next two baselines do not use competence learning for selecting a best classifier or assigning a score to a classifier.

2. Dynamic Ensemble Model (DEM): We create this baseline as follows: We use the correctly classified neighborhood documents to calculate the score for each classifier and then use the scores for doing weighted sum of predicted probabilities of each base classifier as described below:

- **Find the score for each base classifier.** Here we compute the score for each base classifier by applying each base classifier to neighborhood documents. Then the score becomes:

  $Score(B^i) = \#correctly\_classified\_neighbors(B^i)/\#neighbors.$

- **Predict the class label for target document T.** We use weighted sum (similar to our main approach) of predicted probabilities of each base classifier by using scores calculated as above as a weight for each base classifier.

3. Static Ensemble Model (SEM): In this baseline, to make a final prediction, probability for each class label is averaged among all the base classifiers.

4. Majority Vote: We consider a majority vote as another baseline. We predict the document label by a label predicted by a majority of the base classifiers.

5. Bagging[156]: In Bagging, a bag of classifiers, each trained on a random sample of examples from the training set are used to predict the class label for a test example. To predict the target document T, the individual class probabilities assigned by the classifiers in the bag are averaged and the class label with the highest probability is selected. We used BoW classifier for the bagging.

6. META-DES by Cruz et al.[159]: Here, similar to bagging, a pool of classifiers are trained and then the competence or meta-classifier learning is performed to select the competent classifiers out of a pool of classifiers. The majority vote rule is applied over the selected competent classifiers. For META-DES, we used BoW classifiers to generate a pool of classifiers. Note that this baseline includes the competence learning component, but unlike our approach, it uses only one feature type (i.e., BoW).

4.7.2 Experimental Setup

As base classifiers, we experiment with the "bag of words" (BoW) extracted from the entire documents as well as from some portion of the documents, 43 structural features, and the convolutional neural networks (CNN). For the preprocessing step of the BoW, we remove stop words and punctuation, and perform stemming. Moreover, we keep only words that appear in at least 5 documents.

For the traditional base classifiers, we experiment with several machine learning classifiers: Gaussian Naive Bayes (GNB), Multinomial Naive Bayes (MNB), Random Forest (RF), Decision Trees (DT), and Support Vector Machines with a linear kernel (SVM). In addition to these models, we also investigate the performance of CNNs on our task. Our CNNs comprise of mainly two layers (as shown in Figure 4.2): a CNN layer followed by a max pooling

and a fully connected layer for the classification. For the CNN input, we consider a document (partial) as a sequence of words and use pre-trained Word2Vec[39] word embeddings for each word.

Train, Development, Test Splits. From the original datasets of 2,000 PDF files, we divided each dataset into three parts by randomly sampling training set (**Train**), development set (**Dev**), and test set (**Test**) from each dataset. All **Train**, **Dev**, and **Test** follow a similar distribution as the original dataset. Table 4.2 shows the number of positive (+) and negative (-) examples (i.e., documents of interest or not of interest, respectively) in each of the three datasets for which we were able to extract the text (from a given PDF document). For our purpose, to extract the text from the PDF documents, we used PDFBox.[3] The scanned documents and other documents for which the text was not correctly extracted were ignored.

	UNT.edu		Texas.gov		USDA.gov	
Datasets	−	+	−	+	−	+
Train	869	250	981	72	907	121
Dev	290	83	327	24	300	40
Test	290	83	327	24	300	40
Train-2	869	434	981	490	907	453

Table 4.2: *Datasets description.*

Because the original datasets are very skewed (see Section 3), with only around 22%, 7%, and 12% of the PDF documents being part of the positive class (i.e., to be included in a collection), we asked the subject matter experts of each web archive collection to further identify more positive examples. The supplemental positive examples (for each collection) were added to the training set of the corresponding collection. Specifically, for the training set, we sampled from the newly labeled set of positive examples so that the number of negative examples is doubled as compared with the number of positive examples. We denote this set as **Train-2** (see Table 4.2). Note that the test and dev sets of each collection remained the same (i.e., having the original distribution of the data to mimic a real world

[3]http://pdfbox.apache.org/

scenario in which data at test come is fairly skewed).

Note that we studied the performance of our models using other positive to negative data distributions in the training set. However, we found that the models trained on **Train-2** perform better than when we train on other data distributions (e.g., the original or 1:1 data distributions). We estimated this on the development set that we constructed as explained above. Thus, in the next sections, we report the results when we train on **Train-2** (2:1 distribution) and evaluate on **Test** (original distribution).

Moreover, we use the development set also for the hyper-parameter tuning for the classifiers, and for the best classifier selection (which in our experiments was a Random Forest). In experiments, we tuned hyper-parameters for different classifiers as follows: the C parameter in SVM $\in \{0.01, 0.05, 0.1\}$; the number of trees in RF $\in \{20, 23, 25, 27, 30\}$; in CNN, single as well as three types of filters with region sizes $\in \{1, 3, 4, 5, 7\}$; the number of each type of filters in CNN $\in \{100, 128, 200\}$.

Evaluation Measures. To evaluate the performance of various classifiers, we use precision, recall, and F1-score for the positive class. All experiments are repeated three times with a different train/dev/test split obtained using three different random seeds, and the final results are averaged across the three runs. We first discuss the results in terms of the F1-score using bar plots. Then we present all measures: precision, recall, and F1-score, in a table.

4.7.3 Experiments with Base Classifiers

Here, we report the performance of the base classifiers when we train on **Train-2** (2:1 distribution) and evaluate on **Dev** set.

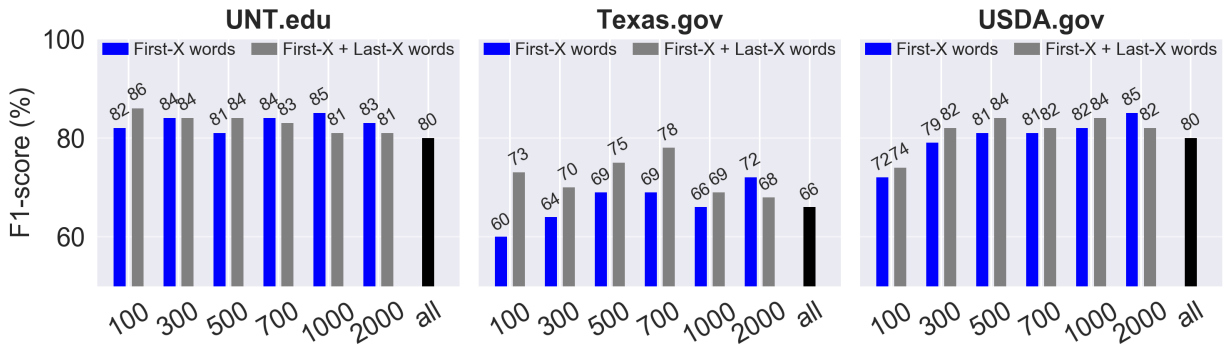


Figure 4.4: *Performance of BoW using different portions of the documents on all three datasets under study.*

The Performance of the BoW Classifier

BoW Performance. First, we compare the performance of the BoW classifiers when we use various portions of the text of the documents with that of the BoW classifiers that use the full text of the documents. For the various portions of the text, we use first X words and first-X words combined with last-X words from each document, where $X \in \{100, 300, 500, 700, 1000, 2000\}$. The results of this set of experiments are shown in Figure 4.4 for all three datasets, UNT.edu, Texas.gov, and USDA.gov, respectively. Random Forest performs best among all classifiers for the BoW features, and hence, we show the results using only Random Forest.

As can be seen from Figure 4.4, on UNT.edu, the BoW that uses the first-100 words combined with last-100 words from each document performs best compared with the performance of the BoW that uses other parts of the documents and achieves a highest F1-score of 0.86. Interestingly, the BoW that uses the entire text of documents performs worse than the BoW that focuses only on specific portions of the text of each document, i.e., the BoW that uses the entire text of the documents achieves a lowest F1-score of 0.80 as compared with 0.86 achieved by BoW that uses only the first-100 + last-100 words from each document. This means that using the entire text of a document introduces redundant or irrelevant features that are not beneficial for the classification task.

On the Texas.gov dataset, it can be seen from Figure 4.4 that the performance of the

BoW classifiers increases as we add more words from the beginning and the end of each document up to 700 words and after that the performance starts decreasing. The BoW classifier that uses the first-700 words combined with last-700 words from each document achieves a highest F1-score of 0.78. On the other hand, the BoW that uses the entire text of documents mostly performs worse than the BoW that focuses only on specific portions of the documents, e.g., the BoW that uses the entire content of the documents achieves an F1-score of 0.66 as compared with the BoW that uses only the first-700 + last-700 words, which achieves an F1-score of 0.78. On Texas.gov, the BoW classifiers that use words from the beginning and ending of the documents generally outperform those that use words only from the beginning of the documents.

On USDA.gov, as can be seen from Figure 4.4, the BoW classifiers that use words from the beginning combined with the ending of the documents (first-X + last-X words) generally outperform the BoW classifiers that use words only from the beginning of documents. These results are similar to those obtained on Texas.gov, although the difference in performance is much smaller compared with Texas.gov. However, interestingly, we notice that the BoW classifier that uses only the first-2000 words performs best and achieves an F1-score of 0.85, which is followed closely by the BoW classifier . As before, the BoW that uses the entire text of documents usually performs worse than the BoW that focuses only on specific portions of each document, e.g., the BoW that uses the entire text of the documents achieves an F1 of 0.80 as compared to 0.85 achieved by BoW on first-2000 words from the documents.

From Figure 4.4, we can also notice that the performance of BoW classifiers on Texas.gov is lower compared with that of classifiers on UNT.edu and USDA.gov, which could be explained by a higher diversity in Texas.gov compared with the other two collections.

Feature selection on the BoW extracted from the entire document text. Next, we show the effect of feature selection on the performance of BoW classifiers that use the full text of the documents. To rank the features and select the top N best features, we use information gain. Figure 4.5 compares the performance of the BoW classifiers that use all features (denoted BoW-all) with that of classifiers that use the top-N selected features

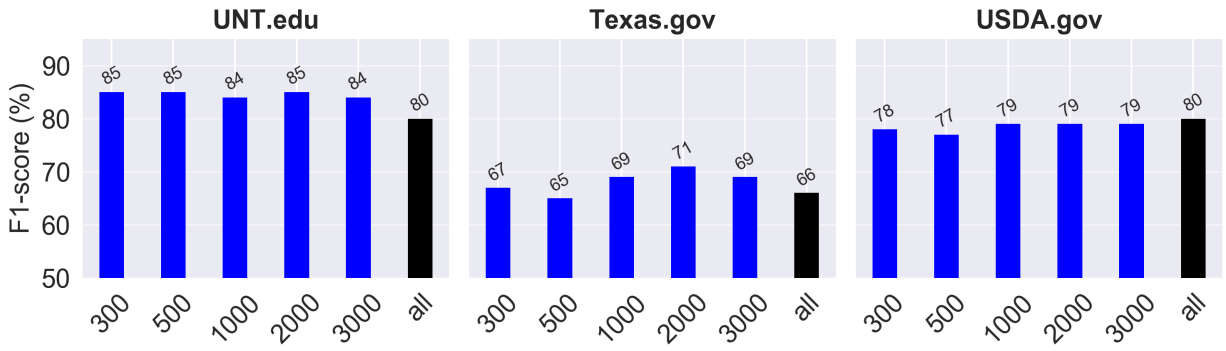


Figure 4.5: *Performance of BoW and its feature selection using the entire content of documents for the BoW encoding, on all three datasets.*

by information gain, for all three datasets, where $N \in \{300, 500, 1000, 2000, 3000\}$. In total, BoW-all has 19733, 19625, and 22255 features for UNT.edu, Texas.gov, and USDA.gov, respectively. From the figure, we notice that performing feature selection improves the performance of BoW-all extracted from the full text of documents for UNT.edu and Texas.gov, whereas the performance decreases slightly on USDA.gov. For example, on UNT.edu, top-300 selected features achieve an F1-score of 0.85 as compared with 0.80 achieved by BoW-all. On Texas.gov, the highest performance is obtained using top-2000 selected features, which achieve a highest F1-score of 0.71 as compared with 0.66 achieved by BoW-all. On USDA.gov, the top-1000, 2000, and 3000 features achieve higher performance as compared to top-300 and top-500 features. Unlike the other two datasets, on the USDA.gov dataset, BoW-all achieves a highest F1 of 0.80 as compared with other top-N selected features. Comparing Figure 4.4 with Figure 4.5, it is interesting to note that, although feature selection improves the performance of BoW-all for UNT.edu and Texas.gov, still the performance of feature selection performed on words from the entire content of documents is not as good as the performance of BoW that uses words from the beginning or the beginning and ending of documents.

Table 4.3 shows the top-30 ranked words using information gain feature selection method on each dataset. For the UNT.edu data, we see tokens that appear to be associated with academic publications such as "data, result, figure, research, or conclusion," which seem to

Dataset	Top-30 Features
UNT.edu	data, al, result, figur, compar, increas, similar, rang, semest, larg, tabl, model, conclusion, research, measur, recent, abstract, exist, show, low, comparison, de, high, usa, observ, doi, base, signific, lack, suggest
Texas.gov	www, texa, program, tx, area, ag, includ, year, inform, system, site, public, nation, contact, result, import, number, manag, reduc, increas, continu, level, servic, plan, base, qualiti, state, work, time, design
USDA.gov	studi, method, research, result, al, effect, potenti, observ, occur, found, measur, speci, water, larg, determin, similar, environ, high, natur, introduc, differ, increas, reduc, analysi, environment, signific, suggest, experi, control, site

Table 4.3: *Top-30 selected features from the BoW (encoded from the entire content of documents) by using information gain.*

match the general scope of this collection as it contains research articles and publications authored by faculty members. The Texas.gov BoW features include tokens that align with publications or other official documents including "texa (texas), program, area, site, nation, or system." These also seem to align very well with the kind of publications selected as being in scope in the dataset. Finally, the USDA.gov BoW selected features include tokens from research and technical publications with tokens such as "study, method, research, result, and effect." There is more overlap between these tokens in USDA.gov and the tokens from the UNT.edu dataset. This suggests that there is some overlap in the kind of content between the two datasets (confirmed by subject matter experts as well).

Next, we explore the following question: *Where are the best performing selected features located in the documents?* To answer this question, we check the overlap between the best performing top-N selected features and the best performing BoW that uses only specific portions of the text of the documents. For all three datasets, we found that all best performing top-N selected features are present in the best performing BoW that uses only specific portions of the document, e.g., on Texas.gov, all top-2000 selected features occur in the BoW that uses the first-700 + last-700 words from each of the documents.

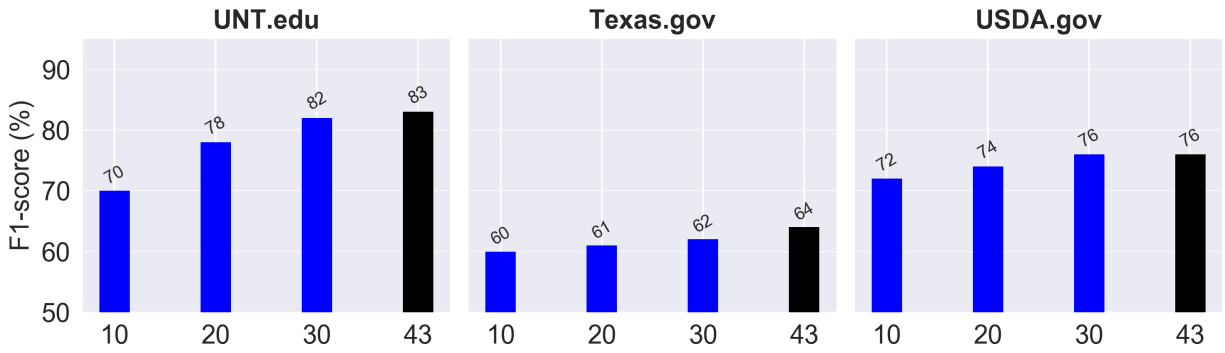


Figure 4.6: *Performance of 43 structural features and its feature selection on all three datasets.*

The performance of structural features and its feature selection

Here, we compare the performance of the 43 structural features with the performance of different top-N selected structural features by information gain. Again, Random Forest performs best compared with any other classifier we experimented with for the structural features and their feature selection. Figure 4.6 shows the performance of the 43 structural features (Str) and the top-N selected features by information gain, for all three datasets. As can be seen from the figure, for UNT.edu and Texas.gov, the performance of the Str classifiers keeps increasing from top-10 features to all 43 features. As expected, on Texas.gov, the performance of Str classifiers is much lower compared with that of Str classifiers on UNT.edu. This is because the Str features are more aligned with academic documents, whereas Texas.gov covers a more diverse set of documents. On USDA.gov, the performance of the Str classifiers keeps increasing from top-10 selected features to top-30 features, and the classifiers corresponding to top-30 and all 43 features perform the same.

Table 4.4 shows the top-30 ranked structural features using the information gain feature selection method on each dataset. Interpreting these structural features is similar to the BoW results discussed above. The UNT.edu shows that the most informative features include those that are closely aligned with the scholarly publications including positionOfThisPaper, refCount, refRatio, and positionOfReferences.

The USDA.gov has similar structural features that key off of the content of the publica-

Dataset	Top-30 Features
UNT.edu	positionOfThisPaper, refCount, refRatio, positionOfReferences, fileSize, tokBeforeRef, references, pgCount, positionOfAbstract, thisPaper, concl, positionOfConcl, strLength, numLines, abstract, numTok, positionOfIntro, ucaseStart, positionOfAck, ack, avgNumWordsPerPage, avgNumLinesPerPage, AckAfterIntro, symbolRatio, spcRatio, symbolStart, lnratio, ucaseRatio, intro, publications
Texas.gov	fileSize, numLines, lnratio, strLength, pgCount, numTok, thisDocument, publications, positionOfIntro, intro, education, positionOfThisPaper, avgNumLinesPerPage, symbolStart, ucaseStart, spcRatio, avgNumWordsPerLine, refRatio, positionOfAck, ack, positionOfConcl, concl, positionOfReferences, positionOfAbstract, tokBeforeRef, references, refCount, avgNumWordsPerPage, ucaseRatio, AckBeforeIntro
USDA.gov	refCount, refRatio, strLength, positionOfThisPaper, pgCount, numTok, positionOfIntro, intro, numLines, positionOfAbstract, positionOfReferences, references, abstract, tokBeforeRef, ucaseStart, fileSize, positionOfConcl, concl, spcRatio, positionOfAck, ack, symbolRatio, ucaseRatio, thisPaper, symbolStart, AckAfterIntro, lnratio, avgNumWordsPerPage, avgNumWordsPerLine, avgNumLinesPerPage

Table 4.4: *Top-30 selected features from the 43 structural features using information gain.*

tions but also start to include more generic features such as strLength, pgCount, and numTok into the most informative features. The Texas.gov is very different, with the most informative structural features being those that are very generic such as fileSize, numLines, lnratio, strLength, and pgCount. This seems to match the content in the datasets where UNT.edu is well focused on scholarly publications, USDA.gov includes both scholarly publications as well as technical reports, and Texas.gov is very broad in the kind of publications included in the collection. Because of this broad range of publications in Texas.gov, it appears that the 43 structural features selected are not being used to their fullest capability for this dataset.

The Performance of the CNN Classifier

Next, we compare the performance of the CNN classifiers when we consider the text from different portions of the documents. Figure 4.7 shows the performance of the CNN classifiers that use word sequences from various portions of the documents, i.e., first X words and first-X words combined with last-X words (where $X \in \{100, 300, 500, 700, 1000\}$).

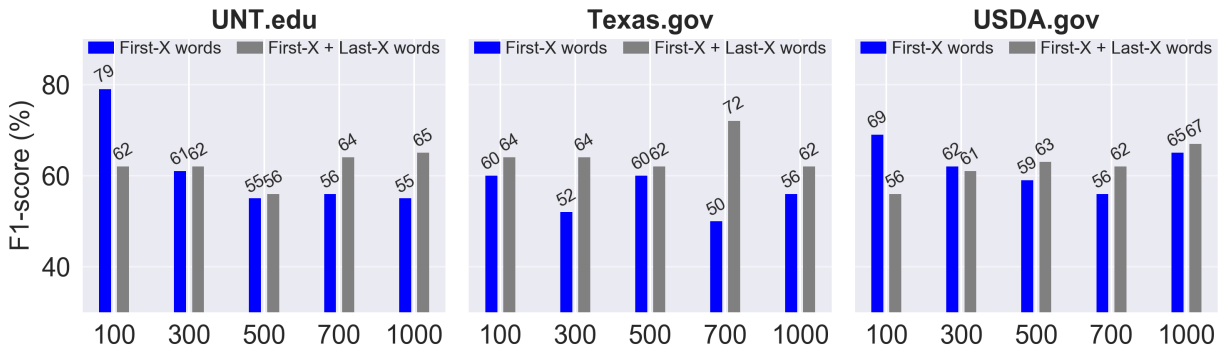


Figure 4.7: *Performance of the CNN using different portions of the documents on different datasets.*

On UNT.edu, it can be seen from Figure 4.7 that the CNN that uses the first-100 words from each of the documents performs best and achieves a highest F1-score of 0.79. Also, on UNT.edu, the CNNs that use words from the beginning and ending of each document generally outperform the CNNs that use words only from the beginning of the documents (except for 100-length sequences). Moreover, we notice the drastic performance gap between the performance of the CNN that uses the first-100 words and the CNNs that use other portions of the documents' text.

On Texas.gov, it can be seen from Figure 4.7 that the CNN classifier that uses the first-700 words combined with the last-700 words performs best and achieves an F1-score of 0.72. For Texas.gov, the CNN classifiers that use words from the beginning and ending portions of documents outperform the CNNs that use words only from the beginning of documents.

On USDA.gov, we can see from Figure 4.7 that the CNN classifier that uses the first-100 words from each document performs best and achieves a highest F1-score of 0.69. The performance of the CNN classifiers that use word sequences from the beginning and ending of documents perform better than those that use only the beginning of documents for word sequence lengths greater than 500. Comparing the results in Figure 4.7 with the previous results, we can notice that the deep learning CNN models perform worse than the Random Forest classifiers that use BoW from various portion of the document text.

	Lib.edu		State.gov		USDA.gov	
	+(%)	All (%)	+ (%)	All (%)	+(%)	All (%)
BoW is correct	78	92	85	96	84	94
Str is correct	84	93	88	93	83	94
CNN is correct	80	90	83	94	78	91
All are correct	63	91	71	97	66	95
All are wrong	4	1.2	2.7	1	5.8	1.4
At least one	96	90	97	89	94	90

Table 4.5: *Exploratory analysis.*

4.7.4 Exploratory Analysis

We perform an exploratory analysis in Table 4.5 to highlight the potential of using our algorithm for selecting best classifier for classifying a document. We predict the class label for a given document by using different individual classifiers and obtained the coverage of the positive class (+ve) and the overall accuracy (All) for the cases when: (1) an individual base classifier is correct, (2) all base classifiers are correct, (3) all base classifiers are wrong, and (4) at least one base classifier is correct. It can be seen from the table that, the last row has the highest value regarding the coverage of the positive class (+ve) among all the datasets and it shows that there is room for improvement for the base classifiers (first three rows), i.e., "At least one" covers 96%, 97%, and 94% of the positive examples as compared with 84% (Str), 88% (Str), and 84% (BoW) for the Lib.edu, State.gov, and USDA.gov, respectively. The large gap between the coverage of the base classifiers (first three rows) and the "At least one" (last row) for the coverage of the positive class (+ve) showcases the potential of combining base classifiers for improving on individual base classifier.

4.7.5 Proposed Model DCSDC vs. Individual Models and Baselines

We contrast the performance of our proposed model DCSDC with that of the three base classifiers (Section 4.4: BoW, CNN, and Str), late fusion of the base classifiers (BoW+Str,

Classifier	UNT.edu				Texas.gov				USDA.gov			
	Pr(+)	Re(+)	F1(+)	Acc.(%)	Pr(+)	Re(+)	F1(+)	Acc.(%)	Pr(+)	Re(+)	F1(+)	Acc.(%)
BoW	0.87	0.78	0.82	92.4	0.64	0.85	0.73	95.7	0.75	0.84	0.79	94.7
Str	0.86	0.84	0.85	93.2	0.50	0.88	0.64	93.3	0.70	0.83	0.76	93.8
CNN	0.75	0.80	0.77	89.5	0.53	0.83	0.65	93.7	0.62	0.78	0.68	91.5
$DCSDC_3$	0.88	0.85	0.87	94.2	0.69	0.88	0.77	96.5	0.77	0.86	0.81	95.2
BoW+Str	**0.94**	0.83	0.88	95.0	0.61	0.89	0.72	95.4	0.74	0.88	0.80	94.8
BoW+CNN	0.83	0.81	0.82	92.1	0.64	0.89	0.74	95.7	0.75	0.84	0.79	94.7
Str+CNN	0.88	**0.86**	0.87	94.2	0.65	0.93	0.77	96.1	0.76	0.85	0.80	95.0
BoW+Str+CNN	0.91	0.85	0.88	94.8	0.69	0.93	0.79	96.6	0.78	0.87	0.82	95.5
$DCSDC_7$	**0.94**	**0.86**	**0.90**	**95.5**	**0.74**	**0.94**	**0.83**	**97.3**	0.81	**0.89**	**0.85**	96.3
KNN	0.85	0.17	0.29	80.9	0.34	0.49	0.35	88.5	0.51	0.12	0.19	88.6
DEM	0.93	0.84	0.88	95.1	0.59	0.88	0.70	95.0	0.74	0.88	0.80	94.9
SEM	0.91	0.85	0.88	94.8	0.69	0.93	0.79	96.6	0.78	0.87	0.82	95.5
Majority $Vote_3$	0.90	0.83	0.86	94.1	0.70	0.88	0.78	96.6	0.76	0.85	0.80	95.0
Majority $Vote_7$	0.91	0.85	0.88	94.8	0.69	0.93	0.79	96.6	0.78	0.87	0.82	95.5
Bagging	0.90	0.78	0.84	92.7	0.67	0.86	0.75	95.9	0.76	0.86	0.80	95.1
META-DES	0.92	0.82	0.86	94.2	0.73	0.85	0.78	96.8	**0.84**	0.88	**0.85**	**96.5**

Table 4.6: *Performance of different features/models on our datasets.*

BoW+CNN, Str+CNN, and BoW+Str+CNN) and six baselines (Section 4.7.1: KNN, DEM, SEM, Majority Vote, Bagging, and META-DES) in terms of all compared measures, precision, recall and F1-score for the positive class, Pr(+), Re(+), and F1(+), and the overall accuracy of the classifier on the **Test** set. $DCSDC_3$ considers only the three base classifiers (BoW, CNN, and Str) and $DCSDC_7$ considers the three base classifiers along with their four late fusions (BoW+Str, BoW+CNN, Str+CNN, and BoW+Str+CNN). For DEM, SEM and Majority Vote, we experiment with considering only three base classifiers as well as base classifiers along with their late fusion, but the performance for DEM and SEM did not change. Majority $Vote_3$ and Majority $Vote_7$ indicate Majority Vote baseline by considering only three base classifiers and base classifiers along with their late fusion, respectively.

As we can see from Table 4.6, the $DCSDC_3$ outperforms the individual base classifiers (BoW, Str, and CNN). Moreover, we can see that the $DCSDC_7$ is the highest performing model across all three datasets in terms of all compared measures except the precision and accuracy on USDA.gov. On the other hand, the performance of the KNN classifier is worst as compared with all other compared classifiers.

On UNT.edu, Str outperforms the other two base classifiers, i.e., Str achieves an F1 of 0.85 as compared with 0.82 and 0.77 achieved by BoW and CNN, respectively. Late fusion

of base classifiers outperforms the corresponding base classifiers, i.e., Str+CNN achieves an F1 of 0.87 as compared with 0.85 and 0.77 achieved by Str and CNN, respectively. KNN, DEM, and both Majority Vote baselines outperform individual base classifiers. $DCSDC_7$ achieves the highest values among all the measures. Furthermore, BoW+Str and Str+CNN also achieve highest precision and highest recall, respectively.

On Texas.gov, BoW outperforms the other two base classifiers, i.e., BoW achieves an F1 of 0.73 as compared with 0.64 and 0.65 achieved by Str and CNN, respectively. Late fusion of base classifiers outperforms the corresponding base classifiers except BoW+Str. All baselines except KNN, and DEM outperform individual base classifiers. $DCSDC_7$ achieves again the highest values overall.

On USDA.gov, BoW outperforms the other two base classifiers, i.e., BoW achieves an F1 of 0.79 as compared with 0.76 and 0.68 achieved by Str and CNN, respectively. Late fusion of base classifiers outperforms the corresponding base classifiers, i.e., Str+CNN achieves an F1 of 0.80 as compared with 0.76 and 0.68 achieved by Str and CNN, respectively. All baselines except KNN outperform individual base classifiers. Surprisingly, META-DES achieves the highest recall, F1 and accuracy. $DCSDC_7$ achieves the highest values for the recall and F1.

4.7.6 Proposed Model DDFC vs. Individual Models and Baselines

We contrast the performance of our proposed model DDFC with three base classifiers (Section 4.4: BoW, CNN, and Str) and seven baselines (Section 4.7.1: KNN, $DCSDC_3$, DEM, SEM, Majority Vote, Bagging, and META-DES) in terms of all compared measures (in %), precision, recall and F1-score for the positive class, Pr(+), Re(+), and F1(+), and the overall accuracy of the classifier on the **Test** set.

Table 4.7 shows the performance of three base classifiers, seven baselines including two previous works, and the proposed approach DDFC. As we can see from the table, META-DES generally outperforms all the base classifiers and other baseline models. However, our proposed approach DDFC that dynamically assigns score to the each base classifier with

Classifier/Model	UNT.edu				Texas.gov				USDA.gov			
	Pr(+)	Re(+)	F1(+)	Acc.	Pr(+)	Re(+)	F1(+)	Acc.	Pr(+)	Re(+)	F1(+)	Acc.
BoW	86.67	77.91	82.02	92.40	64.21	84.72	73.05	95.73	74.71	84.17	78.95	94.71
Str	85.68	83.53	84.56	93.21	50.41	87.50	63.96	93.26	70.23	83.33	76.10	93.82
CNN	75.16	80.32	77.48	89.54	52.78	83.33	64.55	93.73	62.20	77.50	68.29	91.47
KNN	85.05	17.27	28.68	80.88	34.05	48.61	35.25	88.51	51.09	11.67	18.78	88.62
$DCSDC_3$	88.36	85.14	86.71	94.19	69.30	87.50	77.32	96.49	76.76	85.83	80.81	95.20
DEM	93.41	83.94	88.40	95.08	58.95	87.50	70.41	94.97	74.07	87.50	80.19	94.90
SEM	91.46	84.74	87.90	94.82	68.55	**93.06**	78.92	96.58	78.08	86.67	81.97	95.49
Majority Vote	90.09	82.73	86.20	94.10	70.36	87.50	77.94	96.58	75.68	85.00	79.95	95.00
Bagging	89.97	77.91	83.51	92.73	67.42	86.11	75.48	95.90	75.58	85.93	79.94	95.14
META-DES	91.66	81.53	86.13	94.19	72.62	84.72	78.21	96.77	83.90	**87.50**	**85.38**	**96.47**
DDFC	**93.55**	**85.94**	**89.52**	**95.53**	**82.92**	91.68	**86.94**	**98.15**	**84.59**	85.83	85.16	**96.47**

Table 4.7: *Performance (in %) of different features/models on our datasets.*

competence learning, outperforms all three base classifiers and all the baseline methods in terms of F1 (in most cases). Bagging that uses a pool of classifiers based on BoW from a subset of documents outperforms BoW baseline. Moreover, baselines that select from or make use of all the base classifiers ($DCSDC_3$, DEM, SEM, Majority Vote, and META-DES) usually outperform base classifiers except BoW outperforms DEM on State.gov. On the other hand, the performance of the KNN classifier is worst as compared with all other compared classifiers. All baselines except KNN outperform all three base classifiers.

On UNT.edu, DDFC outperforms all other classifiers among all the measures, i.e., DDFC achieves the highest F1 of 89.52%. Among the base classifiers, Str outperforms the other two base classifiers, i.e., Str achives an F1 of 84.56% as compared with 82.02% and 77.48% achieved by BoW and CNN, respectively.

On Texas.gov, DDFC achieves highest values among all the measures except the recall, i.e., DDFC achieves a highest F1 of 0.87. Surprisingly, SEM baseline achieves a highest recall of 93.06%. BoW outperforms the other two baselines, i.e., BoW achieves F1 of 73.05% as compared with 63.96% and 64.55% achieved by Str and CNN, respectively.

On USDA.gov, DDFC achieves a highest value of precision and accuracy of 84.59% and 96.47%, respectively. Moreover DDFC achieves an F1 of 85.16% which is very close to a highest F1 of 85.38% achieved by META-DES. Among the base classifiers, BoW outperforms the other two baselines, i.e., BoW achives an F1 of 78.95% as compared with 76.10% and

68.29% achieved by Str and CNN, respectively.

4.8 Conclusions and Future Directions

In this chapter, we studied different types of features and learning models to accurately distinguish documents of interest for a collection, from web archive data. Experimental results show that BoW features extracted using only some portions of the documents outperform BoW features extracted using the entire content of documents (full text) as well as the top-N selected BoW features, structural features, top-N selected structural features, and a CNN classifier. We found that feature selection done using information gain improved the performance of the BoW classifier. However, our conclusion is that text from specific portions of documents (e.g., the first-X or first-X+last-X number of words from the content of documents) is more useful than the text from the entire content for finding documents of interest to a given collection. Furthermore, we proposed an approach named DDSDC that dynamically selects a classifier to identify if a document is relevant to a collection by dynamically capturing relevant aspects of the given document. Our experimental results show that the DDSDC performs better than the individual classifiers and other strong baseline models for finding documents of interest to a given collection as the collections may include documents of diverse types. Moreover, proposed another competence learning based dynamic classifier selection algorithm named DDFC, and showed that it outperforms strong baselines. Our datasets and code will be made available to the research community to further research in this area.

In the future work, other approaches may produce more powerful models that could be more difficult to interpret or explain. Such models and in depth explorations of deep learning will be tested in the future. Moreover, it will be interesting to explore domain adaptation by training the classifiers on one type of a Web archiving collection and testing on another collection.

Chapter 5

Summary and Discussion

In this chapter, we summarize the the contributions of this work and present future direction in our research.

5.1 Dissertation Summary

Our main goal in this study is to explore keyphrase extraction and its applications to digital libraries. In this study, we have proposed two keyphrase extraction algorithms, CRF-based supervised approach and unsupervised approach KPRank. We also explored integration of different keyphrase extraction models in to CiteSeerX. Moreover, we proposed two search based frameworks for acquiring scientific documents and maintaining the accurate list of researchers' homepages: Scientific Documents Discovery and Researchers' Homepages Discovery. Furthermore, we explored different machine learning and deep learning models for identifying documents in-scope of a collection from Web archives. We proposed two dynamic classifier selection algorithms: Dynamic Classifier Selection for Document Classification (or DCSDC), and Dynamic Decision level Fusion for Document Classification (or DDFC).

The following is a summary of this dissertation:

- **Keyphrase Extraction from Scientific Documents:** In this chapter, we discussed keypharse extraction approaches and the exploration of deployment of keyphrase extraction models in to CiteSeerX digital library. We proposed a CRF based supervised keyphrase extraction approach that utilizes word embeddings as features along with document specific features. We showed that the proposed CRF based model outperforms strong baseline methods. Moreover, we proposed a novel unsupervised graph-based algorithm, KPRank, that incorporates both positional information of the words along with contextual word embeddings for computing a biased PageRank score for each candidate word. Our experimental results on five datasets show that incorporating position information into our biased KPRank model yields better performance compared with a KPRank that does not use the position information. Moreover, we showed that KPRank outperforms strong baseline methods. In the CiteSeerX case study, we proposed to integrate three supervised keyphrase extraction models into CiteSeerX which are more robust than the previously used NP-Chunking method. To evaluate the keyphrase extraction methods from a user perspective, we implemented a voting system on papers' summary pages in CiteSeerX to vote on predicted phrases without showing the model information to reduce potential judgment bias from voters.

- **Applications of Keyphrases/Keywords:** In this chapter, we proposed two search driven approaches for acquiring scientific documents and maintaining a large collection of researcher homepages. Keyphrases or keywords are very useful to formulate queries that can retrieve topically-related articles from the Web. Scholarly digital libraries provide access to scientific publications and comprise useful resources for researchers. Through an experiment using a large collection of $\approx 76,000$ queries (titles + authors names), our scientific documents discovery framework was able to automatically acquire an overall collection of $\approx 267,000$ unique research papers and was able to recover 78% of the original searched titles, i.e., $\approx 34,000$ papers from the $43,496$ original searched titles. To our knowledge, we are the first to interleave Web search and deep learning for researcher homepage identification to build an efficient author homepage

acquisition approach. We deploy our framework in to CiteSeerX for researcher homepage discovery. We show that self-training can be very useful to train deep learning based researcher homepage classifiers using small amount of labeled data along with unlabeled data. Since data annotation is very expensive, we show that human effort can be reduced through self-training, which could be useful when deploying this into another system in future. More, we discovered $12,199$ researcher homepages using $10,000$ paper title queries. This shows the capability of research paper titles for finding researcher homepages. We show the integration of our framework in CiteSeerX for collecting URLs for crawling scientific documents.

- **Document Classification in Web Archiving Collections:** In this chapter, we explore different learning models and feature representations to determine the best performing ones for identifying the documents of interest from the web archived data. Specifically, we study both machine learning and deep learning models and "bag of words" (BoW) features extracted from the entire document or from specific portions of the document, as well as structural features that capture the structure of documents. Moreover, we explore dynamic fusion models to find, on the fly, the model or combination of models that performs best on a variety of document types. We proposed two dynamic classifier selection algorithms: Dynamic Classifier Selection for Document Classification (or DCSDC), and Dynamic Decision level Fusion for Document Classification (or DDFC). We focus our evaluation on three datasets that we created from three different Web archives. Experimental results show that BoW features extracted using only some portions of the documents outperform BoW features extracted using the entire content of documents (full text) as well as the top-N selected BoW features, structural features, top-N selected structural features, and a CNN classifier. We found that feature selection done using information gain improved the performance of the BoW classifier. Our experimental results show that the approach that fuses different models outperforms individual models and other ensemble methods on all three datasets. However, our conclusion is that text from specific portions of documents

(e.g., the first-X or first-X+last-X number of words from the content of documents) is more useful than the text from the entire content for finding documents of interest to a given collection. We show that the DDSDC performs better than the individual classifiers and other strong baseline models. Furthermore, we showed that DDFC also outperforms strong baselines including DDSDC.

5.2 Summary of Contributions

This section presents the contributions of our works in this dissertation:

1. **Keyphrase Extraction from Scientific Documents:**

   - We propose to incorporate word semantics in CRF models for keyphrase extraction through the use of word embeddings. We study the sensitivity of CRFs based on word embedding types, i.e., those pre-trained on Google News as well as those trained on a large collection of ACM research papers. As part of our contributions, we will make available the IDs of our ACM dataset and the word embeddings.
   - We experimentally show that the CRF models that use word embeddings in addition to features extracted from the document itself outperform strong baselines and other previous approaches for keyphrase extraction.
   - We propose KPRank, an unsupervised graph-based algorithm that exploits both the position of words in a document and the contextual word embeddings for computing a biased PageRank score for ranking candidate phrases
   - We show empirically that infusing position information into our biased KPRank model yields better performance compared with its counterpart that does not use the position information. In addition, KPRank with contextual SciBERT embeddings performs better than FastText-based KPRank. Finally, we show that KPRank outperforms many previous unsupervised models.

- We review keyphrase extraction in scholarly digital libraries, using CiteSeerX as a case study. Moreover, we show the development and deployment requirements of the keyphrase extraction models and the maintenance requirements.

2. **Applications of Keyphrases/Keywords:**

   - We propose a novel integrated framework based on search-driven methods to automatically acquire research documents for scientific collections. To our knowledge, we are the first to use "Web Search" based on author names to obtain seed URLs for initiating crawls in an open-access digital library. Moreover, we design a traditional machine learning based novel homepage identification module and adapt existing research on academic document classification, which are crucial components of our framework. We show experimentally that our homepage identification module and the research paper classifier substantially outperform strong baselines.
   - To automatically acquire research documents, we perform a large-scale, first-of-its-kind experiment using $43,496$ research paper titles and $32,816$ author names from Computer and Information Sciences. We compare our framework with two baselines, a breadth-first search crawler and, to the extent possible, the Microsoft Academic. We discuss that our framework does not substitute these systems, but rather they very well complement each other. We compare our framework with two baselines, a breadth-first search crawler and, to the extent possible, the Microsoft Academic. We discuss that our framework does not substitute these systems, but rather they very well complement each other. As part of our contributions, we will make all the constructed datasets available.
   - We propose a search-driven homepage finding approach that uses author names and paper titles to find researcher homepages. To our knowledge, we are the first to use *"paper titles"* as queries to discover researcher homepages. Furthermore, we explore Convolutional Neural Networks (CNNs) for author homepage identification, which is a crucial component in our approach. We conduct a thorough

evaluation of the CNN models trained on both URLs and page content, and show significant improvements in performance over baselines and prior works. Furthermore, we show that self training can improve the performance of the classifier with the small amount of labeled data along with the unlabeled data.

- To discover researcher homepages, we perform a large-scale experiment using author names and paper titles from Computer Science as queries, and show the effectiveness of our approach in discovering a large number of homepages. Finally, as part of our contributions, all resulting datasets for author homepage identification and homepage discovery will be made available to further research in this area. We show the development and deployment requirements of our proposed approach in CiteSeerX and the maintenance requirements.

3. **Document Classification in Web Archiving Collections:**

   - We built three datasets from three different web archives collected by the UNT libraries, each covering different domains: UNT.edu, Texas.gov, and USDA.gov. Each dataset contains the PDF document along with the label indicating whether a document is in scope of a collection or not. We will make these datasets available to further research in this area.

   - We show that BoW classifiers that use only some portion of the documents outperform BoW classifiers that use full text from the entire content of a document, the structural features based classifiers, and the CNN classifier. We also show that feature selection using information gain improves the performance of the BoW classifiers and structural features based classifiers, and present a discussion on the most informative features for each collection.

   - We propose a dynamic classifier selection for document classification (DCSDC) to dynamically select an appropriate classifier to predict the probability of a target document as being in scope of a collection or not. To dynamically select the classifiers, we consider textual similarity along with the structural aspects of the

documents. We show that DCSDC outperforms all the individual feature set models (base classifiers) and other strong baselines.

- We propose a dynamic decision-level fusion for document classification (DDFC) that derives competence features from neighborhood documents and learns a classifier to assign a competence score for each base classifier (BoW, Str, and CNN) in order to fuse them and to predict the probability of a target document as being in scope of a collection or not. To derive the competence features, we consider textual similarity along with the structural aspects of the documents. We show that DDFC outperforms all the individual feature set models (base classifiers) and other strong baselines including DCSDC.

5.3 Future Directions

Below are different future directions for our work:

- For the keyphrases extraction task, our both proposed aproaches, CRF-based supervised approach and unsupervised approach KPRank, can be explored in the other fields in Computer Science such as Computational Linguistics, as well as other scientific domains, such as Biology, Social Science, Political Science, and Material Sciences. Moreover, since these scientific domains do not generally have author-annotated keyphrases, developments of domain adaptation and transfer learning techniques should also be investigated.

- In our CRF-based supervised keyphrase extraction approach, posterior regularization can be integrated.

- For acquiring scientific documents and maintaining the accurate list of researchers' homepages, our both proposed frameworks, Scientific Documents Discovery and Researchers' Homepages Discovery, can be applied to other domains, and study the integration of topic classification.

- For identifying documents in-scope of a collection from the Web archived data, other approaches may produce more powerful models that could be more difficult to interpret or explain. Such models and in depth explorations of deep learning will be tested in the future. Moreover, it will be interesting to explore domain adaptation by training the classifiers on one type of a Web archiving collection and testing on another collection.

Bibliography

[1] Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. Content-based citation recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 238–251, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1022. URL https://www.aclweb.org/anthology/N18-1022.

[2] Ding Zhou, Shenghuo Zhu, Kai Yu, Xiaodan Song, Belle L. Tseng, Hongyuan Zha, and C. Lee Giles. Learning multiple graphs for document recommendations. In *Proc. of WWW '08*, 2008.

[3] Cornelia Caragea, Adrian Silvescu, Prasenjit Mitra, and C. Lee Giles. Can't see the forest for the trees?: a citation recommendation system. In *13th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '13, Indianapolis, IN, USA, July 22 - 26, 2013*, pages 111–114, 2013.

[4] Krisztian Balog and Maarten De Rijke. Determining expert profiles (with an application to expert finding). In *IJCAI*, 2007.

[5] Sujatha Das Gollapalli, Prasenjit Mitra, and C. Lee Giles. Similar researcher search in academic environments. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '12, Washington, DC, USA, June 10-14, 2012*, pages 167–170, 2012.

[6] Cornelia Caragea, Florin Adrian Bulgarov, and Rada Mihalcea. Co-training for topic classification of scholarly data. In *Proceedings of the 2015 Conference on Empirical*

Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 2357–2366, 2015.

[7] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, 2003.

[8] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592. Association for Computational Linguistics, 2017.

[9] Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.103. URL https://www.aclweb.org/anthology/2020.acl-main.103.

[10] Krutarth Patel and Cornelia Caragea. Exploring word embeddings in crf-based keyphrase extraction from research papers. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 37–44, 2019.

[11] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1435–1446, 2014. URL http://aclweb.org/anthology/D/D14/D14-1150.pdf.

[12] Kazi Saidul Hasan and Vincent Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *COLING*, pages 365–373, 2010.

[13] Kazi Saidul Hasan and Vincent Ng. Automatic keyphrase extraction: A survey of the state of the art. In *ACL*, pages 1262–1273, June 2014.

[14] Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. In *SemEval@ACL*, pages 546–555, 2017.

[15] Amjad Abu-Jbara and Dragomir Radev. Coherent citation-based summarization of scientific papers. In *ACL: HLT*, pages 500–509, 2011. ISBN 978-1-932432-87-9.

[16] Vahed Qazvinian, Dragomir R. Radev, and Arzucan Özgür. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 895–903, 2010.

[17] Peter D Turney. Coherent keyphrase extraction via web mining. *arXiv preprint cs/0308033*, 2003.

[18] Wen-tau Yih, Joshua Goodman, and Vitor R. Carvalho. Finding advertising keywords on web pages. In *WWW'06*, pages 213–222, 2006. ISBN 1-59593-323-9.

[19] Gábor Berend. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170, 2011.

[20] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Automatic keyphrase extraction from scientific articles. *Language resources and evaluation*, 47 (3):723–742, 2013.

[21] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*, pages 216–223, 2003.

[22] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, pages 855–860, 2008.

[23] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, pages 404–411, 2004.

[24] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI*, pages 668–673, 1999.

[25] Jefferson Bailey. https://twitter.com/jefferson_bail/status/867808876917178368, May 2017.

[26] HathiTrust. Hathitrust statistics information. https://www.hathitrust.org/statistics_info, 2017.

[27] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2), September 2002.

[28] Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. Researcher homepage classification using unlabeled data. In *WWW*, 2013.

[29] Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. Incorporating expert knowledge into keyphrase extraction. In *AAAI*, pages 3180–3187, 2017.

[30] Pinaki Bhaskar, Kishorjit Nongmeikapam, and Sivaji Bandyopadhyay. Keyphrase extraction in scientific articles: A supervised approach. In *COLING*, pages 17–24, Mumbai, India, December 2012.

[31] Chengzhi Zhang, Huilin Wang, Yao Liu, Dan Wu, Yi Liao, and Bo Wang. Automatic keyword extraction from documents using conditional random fields. *JCIS*, 4(3):1169–1180, 2008.

[32] Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *NAACL*, pages 634–639, 2018.

[33] Corina Florescu and Cornelia Caragea. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *ACL*, pages 1105–1115, Vancouver, Canada, July 2017.

[34] Krutarth Patel, Cornelia Caragea, Jian Wu, and C Lee Giles. Keyphrase extraction in scholarly digital library search engines. In *International Conference on Web Services*, pages 179–196. Springer, 2020.

[35] Krutarth Patel, Cornelia Caragea, and Sujatha Das Gollapalli. On the use of web search to improve scientific collections. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 174–183, 2020.

[36] Krutarth Patel, Cornelia Caragea, Doina Caragea, and C Lee Giles. Author homepage discovery in citeseerx. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[37] Krutarth Patel, Cornelia Caragea, Mark E Phillips, and Nathaniel T Fox. Identifying documents in-scope of a collection from web archives. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, pages 167–176, 2020.

[38] Krutarth Patel, Cornelia Caragea, and Mark Phillips. Dynamic classification in web archiving collections. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1459–1468, 2020.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[40] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394, 2010.

[41] Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. Automatic keyword extraction on twitter. In *ACL and IJCNLP*, 2015.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[43] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[44] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[45] Hung-Hsuan Chen, Pucktada Treeratpituk, Prasenjit Mitra, and C Lee Giles. Csseer: an expert recommendation system based on citeseerx. In *JCDL*, pages 381–382, 2013.

[46] Thuy Dung Nguyen and Min-Yen Kan. Keyphrase extraction in scientific publications. In *Asian Digital Libraries*, pages 317–326, 2007.

[47] Lucas Sterckx, Cornelia Caragea, Thomas Demeester, and Chris Develder. Supervised keyphrase extraction as positive unlabeled learning. In *EMNLP*, pages 1924–1929, 2016.

[48] Olena Medelyan, Eibe Frank, and Ian H Witten. Human-competitive tagging using automatic keyphrase extraction. In *EMNLP*, pages 1318–1327, 2009.

[49] Patrice Lopez and Laurent Romary. Humb: Automatic key term extraction from scientific articles in grobid. In *SemEval*, pages 248–251, 2010.

[50] Florin Bulgarov and Cornelia Caragea. A comparison of supervised keyphrase extraction models. In *WWW*, pages 13–14, 2015.

[51] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *EMNLP*, pages 257–266, 2009.

[52] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In *Advances in Artificial Intelligence*, pages 40–52. Springer, 2000.

[53] Yongzheng Zhang, Evangelos Milios, and Nur Zincir-Heywood. A comparative study on key phrase extraction methods in automatic web site summarization. *JDIM*, 5(5): 323, 2007.

[54] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. Extracting key terms from noisy and multitheme documents. In *WWW*, pages 661–670, 2009.

[55] Shibamouli Lahiri, Sagnik Ray Choudhury, and Cornelia Caragea. Keyword and keyphrase extraction using centrality measures on collocation networks. *CoRR*, abs/1401.6571, 2014. URL http://arxiv.org/abs/1401.6571.

[56] Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *ACL*, pages 620–628, 2009.

[57] Nedelina Teneva and Weiwei Cheng. Salience rank: Efficient keyphrase extraction with topic modeling. In *ACL*, volume 2, pages 530–535, 2017.

[58] Ido Blank, Lior Rokach, and Guy Shani. Leveraging the citation graph to recommend keywords. In *RecSys*, pages 359–362, 2013.

[59] Sujatha Das Gollapalli and Cornelia Caragea. Extracting keyphrases from research papers using citation networks. In *AAAI*, pages 1629–1635, 2014.

[60] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. Automatic keyphrase extraction via topic decomposition. In *EMNLP*, pages 366–376, 2010.

[61] Juan Martinez-Romo, Lourdes Araujo, and Andres Duque Fernandez. Semgraph: Extracting keyphrases following a novel semantic graph-based approach. *JASIST*, 67(1): 71–82, 2016.

[62] Samhaa R El-Beltagy and Ahmed Rafea. Kp-miner: Participation in semeval-2. In *SemEval*, pages 190–193, 2010.

[63] Soheil Danesh, Tamara Sumner, and James H Martin. Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. *Lexical and Computational Semantics*, page 117, 2015.

[64] Eytan Adar and Srayan Datta. Building a scientific concept hierarchy database (schbase). In *ACL*, pages 606–615, 2015.

[65] Wei Fan, Huan Liu, Suge Wang, Yuxiang Zhang, and Yaocheng Chang. Extracting keyphrases from research papers using word embeddings. In *Advances in Knowledge Discovery and Data Mining*, pages 54–67, 2019. ISBN 978-3-030-16142-2.

[66] Rui Wang, Wei Liu, and Chris McDonald. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, page 39, 2014.

[67] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. Simple unsupervised keyphrase extraction using sentence embeddings. In *CoNLL*, pages 221–229, 2018.

[68] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. Keyphrase extraction using deep recurrent neural networks on twitter. In *EMNLP*, pages 836–845, 2016.

[69] Jishnu Ray Chowdhury, Cornelia Caragea, and Doina Caragea. Keyphrase extraction from disaster-related tweets. In *The World Wide Web Conference*, WWW '19, pages 1555–1566, 2019. ISBN 978-1-4503-6674-8.

[70] Isabelle Augenstein and Anders Søgaard. Multi-task learning of keyphrase boundary classification. In *ACL*, volume 2, pages 341–346, 2017.

[71] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep keyphrase generation. In *ACL*, pages 582–592, 2017.

[72] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi

Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[73] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. Keyphrase generation with correlation constraints. In *EMNLP*, pages 4057–4066, 2018.

[74] Hai Ye and Lu Wang. Semi-supervised learning for neural keyphrase generation. In *EMNLP*, pages 4142–4153, 2018.

[75] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. Title-guided encoding for keyphrase generation. *CoRR*, abs/1808.08575, 2018.

[76] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, pages 1064–1074, Berlin, Germany, August 2016. URL http://www.aclweb.org/anthology/P16-1101.

[77] Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. Empower Sequence Labeling with Task-Aware Neural Language Model. In *AAAI*, 2018.

[78] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[79] Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *WWW*, pages 2551–2557. ACM, 2019.

[80] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[81] Marina Litvak and Mark Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24, 2008.

[82] Xin Jiang, Yunhua Hu, and Hang Li. A ranking approach to keyphrase extraction. In *SIGIR*, pages 756–757, 2009.

[83] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. *Technical report, Standford Digital Library Technologies Project*, 1998.

[84] Taher H Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, pages 784–796, 2003.

[85] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[86] Yi-fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen. Domain-specific keyphrase extraction. In *CIKM*, pages 283–284, 2005.

[87] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI*, pages 668–673, 1999.

[88] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. SemEval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles. In *SemEval*, pages 21–26, 2010.

[89] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. Large dataset for keyphrases extraction. Technical report, University of Trento, 2009.

[90] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC*, pages 45–50, May 2010.

[91] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *IJCNLP*, pages 543–551, 2013.

[92] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA*, pages 89–98, 1998.

[93] Jian Wu, Kyle Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Alexander Ororbia, Douglas Jordan, and C. Lee Giles. CiteSeerX: AI in a digital library search engine. In *AAAI*, pages 2930–2937, 2014.

[94] P. Teregowda, B. Urgaonkar, and C. L. Giles. Cloud 2010. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 115–122, 2010.

[95] Kyle Williams, Jian Wu, Sagnik Ray Choudhury, Madian Khabsa, and C. Lee Giles. Scholarly big data information extraction and integration in the citeseerx digital library. *IIWeb*, pages 68–73, 2014.

[96] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246. ACM, 2015.

[97] Jian Wu, Bharath Kandimalla, Shaurya Rohatgi, Athar Sefid, Jianyu Mao, and C. Lee Giles. Citeseerx-2018: A cleansed multidisciplinary scholarly big dataset. In *IEEE Big Data*, pages 5465–5467, 2018.

[98] Hui Han, C. Lee Giles, Eren Manavoglu, Hongyuan Zha, Zhenyue Zhang, and Edward A. Fox. Automatic document metadata extraction using support vector machines. In *JCDL*, pages 37–48. IEEE, 2003.

[99] Isaac Councill, C Lee Giles, and Min-Yen Kan. ParsCit: an open-source CRF reference string parsing package. In *LREC*, volume 8, pages 661–667, 2008.

[100] Cornelia Caragea, Jian Wu, Sujatha Das Gollapalli, and C. Lee Giles. Document type classification in online digital libraries. *Innovative Applications of Artificial Intelligence (IAAI)*, 2016.

[101] Pucktada Treeratpituk and C. Lee Giles. Disambiguating authors in academic publications using random forests. In *JCDL*, pages 39–48. ACM, 2009.

[102] Athar Sefid, Jian Wu, Allen C. Ge, Jing Zhao, Lu Liu, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. Cleaning noisy and heterogeneous metadata for record linking across scholarly big datasets. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 9601–9606, 2019.

[103] Grobid. https://github.com/kermitt2/grobid, 2008–2020.

[104] Jian Wu, Jason Killian, Huaiyu Yang, Kyle Williams, Sagnik Ray Choudhury, Suppawong Tuarob, Cornelia Caragea, and C. Lee Giles. Pdfmef: A multi-entity knowledge extraction framework for scholarly documents and semantic search. In *K-CAP*, pages 13:1–13:8. ACM, 2015.

[105] Huajing Li, Isaac G. Councill, Levent Bolelli, Ding Zhou, Yang Song, Wang-Chien Lee, Anand Sivasubramaniam, and C. Lee Giles. Citeseer^{x}: a scalable autonomous scientific digital library. *InfoScale*, 2006.

[106] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. *KDD*, 2008.

[107] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. Title-guided encoding for keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275, 2019.

[108] Marina Litvak and Mark Last. Graph-Based Keyword Extraction for Single-Document Summarization. In *Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization*, MMIES '08, pages 17–24, Manchester, United Kingdom, 2008. ISBN 978-1-905593-51-4.

[109] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006.

[110] Gautam Pant, Kostas Tsioutsiouliklis, Judy Johnson, and C Lee Giles. Panorama: extending digital libraries with topical crawlers. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 142–150. ACM, 2004.

[111] Ziming Zhuang, Rohit Wagle, and C Lee Giles. What's there and what's not?: focused crawling for missing documents in digital libraries. In *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*, pages 301–310. IEEE, 2005.

[112] David Carmel, Elad Yom-Tov, and Haggai Roitman. Enhancing digital libraries using missing content analysis. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '08, pages 1–10, 2008.

[113] Sven E Hug and Martin P Brändle. The coverage of microsoft academic: Analyzing the publication output of a university. *arXiv preprint arXiv:1703.05539*, 2017.

[114] Anne-Wil Harzing and Satu Alakangas. Microsoft academic: is the phoenix getting wings? *Scientometrics*, 110(1):371–383, 2017.

[115] Wensi Xi, Edward Fox, Roy Tan, and Jiang Shu. Machine learning approach for homepage finding task. In *String Processing and Information Retrieval*, pages 169–174. Springer, 2002.

[116] Trystan Upstill, Nick Craswell, and David Hawking. Query-independent evidence in home page finding. *ACM Trans. Inf. Syst.*, 2003.

[117] Yuxin Wang and Keizo Oyama. Web page classification exploiting contents of surrounding pages for building a high-quality homepage collection. *ICADL*, 4312:515–518, 2006.

[118] Jie Tang, Duo Zhang, and Limin Yao. Social network extraction of academic researchers. In *ICDM*, 2007.

[119] Sujatha Das Gollapalli, Prasenjit Mitra, and C. Lee Giles. Learning to rank homepages for researcher name queries. In *EOS Workshop at SIGIR*, 2011.

[120] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2), February 2009.

[121] Junting Ye, Yanan Qian, and Qinghua Zheng. Plidminer: A quality based approach for researcher's homepage discovery. In *Asia Information Retrieval Symposium*, pages 199–210. Springer, 2012.

[122] In-Su Kang, Pyung Kim, Seungwoo Lee, Hanmin Jung, and Beom-Jong You. Construction of a large-scale test set for author disambiguation. *Information Processing & Management*, 47(3):452–465, 2011.

[123] Bo Long, Philip S Yu, and Zhongfei Zhang. A general model for multiple view unsupervised learning. In *SIAM*, pages 822–833, 2008.

[124] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.

[125] Xiao-Yuan Jing, Fei Wu, Xiwei Dong, Shiguang Shan, and Songcan Chen. Semi-supervised multi-view correlation feature learning with application to webpage classification. In *AAAI*, 2017.

[126] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618, 9780262035613.

[127] Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. Using unlabeled data to improve researcher homepage classification. In *Transactions on the Web*, 2015.

[128] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL, 2014. URL http://aclweb.org/anthology/D/D14/D14-1181.pdf.

[129] Jiapeng Zhao, Tingwen Liu, and Jinqiao Shi. Improving academic homepage identification from the web using neural networks. In *ICCS*, pages 551–558. Springer, 2019.

[130] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *WWW*, 1999.

[131] Cristiano Mesquita Garcia, Armando Honorio Pereira, and Denilson Alves Pereira. A framework to collect and extract publication lists of a given researcher from the web. *JWET*, 12(3):234–252, 2017.

[132] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 2009.

[133] Thorsten Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, 2002.

[134] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[135] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[136] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12(Aug): 2493–2537, 2011.

[137] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11): 39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748.

[138] Amanda Spink and Bernard J Jansen. *Web search: Public searching of the Web*, volume 6. Springer Science & Business Media, 2004. ISBN 9781402022692.

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

[140] Steve Lawrence. Free online availability substantially increases a paper's impact. In *Nature*, 411 (6837), pages 521–521, 2001.

[141] José-Luis Ortega-Priego, Isidro F. Aguillo, and José Antonio Prieto-Valverde. Longitudinal study of contents and elements in the scientific web environment. *Journal of Information Science*, 32(4), 2006.

[142] International Internet Preservation Consortium. Members. http://netpreserve.org/about-us/members, 2017.

[143] Archive-It. Archive-it homepage. https://archive-it.org/, 2017.

[144] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. *Text databases and document management: Theory and practice*, 5478: 78–102, 2001.

[145] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1), 2002.

[146] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[147] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[148] Mark Phillips and Kathleen Murray. Improving access to web archives through innovative analysis of pdf content. In *Archiving Conference*, pages 186–192. Society for Imaging Science and Technology, 2013.

[149] James A Jacobs. Born-digital us federal government information: Preservation and access. *Leviathan: Libraries and Government Information in the Age of Big Data*, 2014.

[150] Alexander C Nwala, Michele C Weigle, and Michael L Nelson. Bootstrapping web archive collections from social media. In *Proceedings of the 29th on Hypertext and Social Media*, pages 64–72. ACM, 2018.

[151] Sawood Alam, Fateh ud din B Mehmood, and Michael L Nelson. Improving accessibility of archived raster dictionaries of complex script languages. In *Proceedings of the JCDL*, pages 47–56, 2015.

[152] Sawood Alam, Michael L Nelson, Herbert Van de Sompel, Lyudmila L Balakireva, Harihar Shankar, and David SH Rosenthal. Web archive profiling through cdx summarization. *International Journal on Digital Libraries*, 17(3):223–238, 2016.

[153] Yasmin AlNoamany, Michele C Weigle, and Michael L Nelson. Generating stories from archived collections. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 309–318. ACM, 2017.

[154] Mohamed Aturban, Sawood Alam, Michael Nelson, and Michele Weigle. Archive assisted archival fixity verification framework. In *JCDL*, pages 162–171. IEEE, 2019.

[155] Chase Dooley and Grace Thomas. The library of congress web archives: Dipping a toe in a lake of data. https://blogs.loc.gov/thesignal/2019/01/the-library-of-congress-web-archives-dipping-a-toe-in-a-lake-of-data/, 2019.

[156] Marina Skurichina and Robert PW Duin. Bagging for linear classifiers. *Pattern Recognition*, 31(7):909–930, 1998.

[157] Michał Woźniak, Manuel Graña, and Emilio Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17, 2014.

[158] Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.

[159] Rafael Cruz, Robert Sabourin, and George Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41, 05 2018.

[160] Rafael Cruz, Robert Sabourin, George Cavalcanti, and Tsang Ing Ren. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, 48, 05 2015.

[161] Paulo Rodrigo Cavalin, Robert Sabourin, and Ching Y. Suen. Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, 22: 673–688, 2011.

[162] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *2010 IEEE Computer society conference on computer vision and pattern recognition*, pages 902–909. IEEE, 2010.

[163] Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomput.*, 174(PA):50–59, January 2016. ISSN 0925-2312.

[164] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In *Proceedings of the 31st International Conf. on ML*, volume 32, 22–24 Jun 2014.

[165] Tom Zahavy, Abhinandan Krishnan, Alessandro Magnani, and Shie Mannor. Is a picture worth a thousand words? A deep multi-modal architecture for product classification in e-commerce. In *AAAI*. AAAI Press, 2018.

[166] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[167] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proc. of the 10th ECML*, pages 137–142, 1998.

[168] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI*, 1999.

[169] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM*, CIKM '98, pages 148–155, 1998.

[170] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[171] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge university press, 2008.

[172] Mark Craven, Johan Kumlien, et al. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86, 1999.

[173] Ajith Kodakateri Pudhiyaveetil, Susan Gauch, Hiep Luong, and Josh Eno. Conceptual recommender system for citeseerx. In *RecSys*, pages 241–244. ACM, 2009.

[174] Tong Zhou, Yi Zhang, and Jianguo Lu. Classifying computer science papers. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016.

[175] Yaakov HaCohen-Kerner, Avi Rosenfeld, Maor Tzidkani, and Daniel Nisim Cohen. Classifying papers from different computer science conferences. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar R. Zaïane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications, 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I*, volume 8346 of *Lecture Notes in Computer Science*, pages 529–541. Springer, 2013. doi: 10.1007/978-3-642-53914-5_45. URL https://doi.org/10.1007/978-3-642-53914-5_45.

[176] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[177] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[178] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[179] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[180] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[181] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[182] Wenpeng Yin and Hinrich Schütze. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*, 2016.