THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

ENTITY EXTRACTION FROM UNSTRUCTURED DATA ON THE WEB

Dat Tan Huynh

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in December 2014

School of Information Technology & Electrical Engineering

Abstract

A large number of web pages contain information about entities in lists where the lists are represented in textual form. Textual lists contain implicit records of entities. However, the field values of such records cannot easily be separated or extracted by automatic processes. This, therefore, remains a challenging research problem in the literature. Previous studies in the literature relied mainly on probabilistic graph-based models to capture the attributes and the likely structures of implicit records in a list. However, one of the important limitations of existing methods is that the structures of the records in input lists were implicitly encoded via training data which was manually created. This thesis aims to investigate novel techniques to acquire automatically information about entities from implicit records embedded in textual lists on the web.

This thesis introduces a self-supervised learning framework which exploits both existing data in a knowledge base and the structural similarity between sequences in lists to build an extraction model automatically. In the proposed framework, initial labels for candidate field values are created and assigned to generate label sequences. Then, the structure of implicit records is captured via a graphical model to assign unmatched labels and rectify mismatched labels. As a result of which, the process of entity extraction from lists can be completely unsupervised and automated without user intervention. In order to attain that outcome, we address three substantive research problems that need to be solved in this thesis.

Firstly, the text segments in input lists need to be assigned labels precisely so that their statistical information is then used to build an extraction model. However, previous studies have not considered completely both the format and content of field values when performing the text segmentation and assigning labels. By viewing the problem of assigning labels for text segments as the problem of membership checking in set theory, we identify and propose a dyadic representation of semantic relations between a text segment and an attribute by using its extensional and intensional representations. We incorporate those representations to define a novel format-enhanced labelling technique to assign labels for text segments.

Secondly, the labels of identical concepts with differing sequences in an input list are

often located in similar positions but the positions of the labels may vary somewhat in different sequences. However, until this point, there has been no information extraction system designed to capture the distribution of labels in differing positions in order to enhance extraction results. To capture the positional information of labels, we are proposing a proximity-based positional model, which is combined with a sequential model to improve the quality of the label-refinement phase in our framework.

Thirdly, in order to reduce dependence on the overlap between knowledge bases and input lists, we exploit structural similarity between text segments and sequences in the input lists, and devise a structure-based similarity and data shifting-alignment technique to align text segments into groups before their labels are revised by a graphical model. By the proposed technique, we can reduce the dependency on the overlap between knowledge bases and input lists whilst maintaining high performance of extraction model.

Initially experimental results demonstrate that our proposed techniques perform well when compared to the state-of-the-art method. We hope that the results presented in this thesis contribute to efforts on the extraction of information about entities in textual lists. Additionally, they contribute towards forthcoming research on the synthesis of information from different lists, and the provision of reasoning capacity by which to detect new relationships between entities drawn from raw lists on the web.

Declaration by Author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my research higher degree candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the General Award Rules of The University of Queensland, immediately made available for research and study in accordance with the *Copyright Act 1968*.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis.

Publications during candidature

- Dat Tan Huynh and Xiaofang Zhou, "Automatic Information Extraction from Lists on the Web", submitted to *Journal of the Association for Information Systems*, 2014.

- Dat Tan Huynh, Jiajie Xu, Shazia Sadiq and Xiaofang Zhou, "Exploiting Structural Similarity for Automatic Information Extraction from Lists", *The 14th International Conference on Web Information System Engineering* (WISE 2013), pages 202–215, 2013.

- Dat Tan Huynh and Xiaofang Zhou, "Exploiting a Proximity-based Positional Model to Improve the Quality of Information Extraction by Text Segmentation", *The 24th Australasian Database Conference* (ADC 2013), pages 23–32, 2013.

- Wen Hua, Dat Tan Huynh, Saeid Hosseini, Jiaheng Lu, and Xiaofang Zhou, "Information Extraction From Microblogs: A Survey", *International Journal of Software and Informatics*, 6(4), pages 495–522, 2013.

- Dat Tan Huynh and Wen Hua, "Self-Supervised Learning Approach for Extracting Citation Information on the Web", *The 14th Asia-Pacific Web Conference* (APWEB 2012), pages 719–726, 2012.

Publications included in this thesis

- Dat Tan Huynh and Xiaofang Zhou, "Automatic Information Extraction from Lists on the Web", submitted to *Journal of the Association for Information Systems*, 2014.

Contributor	Statement of contribution
Author Dat Tan Huynh (Candidate)	Defined the problem with motivations (80%) Designed or gave ideas on solutions (100%) Designed and conducted the experiments (100%) Wrote the paper (100%) Revised and proofread the paper (80%)
Author Xiaofang Zhou	Defined the problem with motivations (20%) Gave comments and proofread the paper (20%)

- Dat Tan Huynh, Jiajie Xu, Shazia Sadiq and Xiaofang Zhou, "Exploiting Structural Similarity for Automatic Information Extraction from Lists", *The 14th International Conference on Web Information System Engineering* (WISE 2013), pages 202–215, 2013.

Contributor	Statement of contribution
Author Dat Tan Huynh (Candidate)	Defined the problem with motivations (80%) Designed or gave ideas on solutions (100%) Designed and conducted the experiments (100%) Wrote the paper (100%) Revised and proofread the paper (75%)
Author Jiajie Xu	Revised and proofread the paper (5%)
Author Shazia Sadiq	Revised and proofread the paper (5%)
Author Xiaofang Zhou	Defined the problem with motivations (20%) Gave comments and proofread the paper (15%)

- Dat Tan Huynh and Xiaofang Zhou, "Exploiting a Proximity-based Positional Model to Improve the Quality of Information Extraction by Text Segmentation", *The 24th Australasian Database Conference* (ADC 2013), pages 23–32, 2013.

Contributor	Statement of contribution
Author Dat Tan Huynh (Candidate)	Defined the problem with motivations (80%) Designed or gave ideas on solutions (100%) Designed and conducted the experiments (100%) Wrote the paper (100%) Revised and proofread the paper (80%)
Author Xiaofang Zhou	Defined the problem with motivations (20%) Gave comments and proofread the paper (20%)

- Wen Hua, Dat Tan Huynh, Saeid Hosseini, Jiaheng Lu, and Xiaofang Zhou, "Information Extraction From Microblogs: A Survey", *International Journal of Software and Informatics*, 6(4), pages 495–522, 2013.

Contributor	Statement of contribution
Author Wen Hua	Defined the problem with motivations (80%) Wrote the paper (65%) Revised and proofread the paper (60%)
Author Dat Tan Huynh (Candidate)	Wrote two sections on information extraction on web pages and micro-blogs (30%) Revised and proofread the paper (15%)
Author Saeid Hosseini	Augmented some details of information extraction on microblogs (5%) Revised and proofread the paper (5%)
Author Jiaheng Lu	Revised and proofread the paper (5%)
Author Xiaofang Zhou	Defined the problem with motivations (20%) Gave comments and proofread the paper (15%)

- Dat Tan Huynh and Wen Hua, "Self-Supervised Learning Approach for Extracting Citation Information on the Web", *The 14th Asia-Pacific Web Conference* (APWEB

2012), pages 719–726, 2012.

Contributor	Statement of contribution
Author Dat Tan Huynh (Candidate)	Defined the problem with motivations (100%) Designed or gave ideas on solutions (100%) Designed and conducted the experiments (100%) Wrote, revised, and proofread the paper (90%)
Author Wen Hua	Revised and proofread the paper (10%)

Contributions by others to the thesis

None.

Statement of parts of the thesis submitted to qualify for the award of another degree

None.

Acknowledgments:

"When drinking water, think of its source"

Vietnamese proverb.

I would like to express my sincere gratitude to my principle advisor, Professor Xiaofang Zhou, for his kindness, advice and support throughout my PhD course. I always remember several discussions between us to identify research issues in the first stage of my study. From those discussions, I learnt a lot of research skills in my study. The questions which Professor Xiaofang Zhou frequently asked me in our weekly meetings as well as his comments on my papers trained me a lot about how to identify and address a research problem, how to review critically previous studies, as well as clarify, consolidate and develop ideas from a problem. Gradually, they made me more confident and independent in doing research. They definitely have a deep impact on my professional career in conducting research in the future.

I am also grateful to Professor Tru Hoang Cao, the supervisor of my master thesis in Vietnam National University - Ho Chi Minh City University of Technology. He was the person who introduced me to the life of conducting research, guided and furnished me with basic skills in doing research during the time I worked with him as a research assistant in the university.

I would like to thank Vietnamese government for awarding me an Overseas Postgraduate Research scholarship and The University of Queensland for a University of Queensland Postgraduate Research scholarship. My PhD program would be unfeasible if there were no those financial supports.

Eventually, this special milestone makes me think about my parents, my wife and my children. Neither this thesis nor anything else can be done without the great support of my beloved family. Indeed, my parents and my wife encouraged me a lot when I was stressed and got stuck with ideas and papers. There were several times when their encouragements gave me a spiritual strength to make a big difference between giving up and trying again in my study. I'm grateful to them and I believe that my words are not enough to convey my gratitude to them.

Keywords

Information extraction, entity extraction, list extraction, self-supervised learning, positional model, graphical model, proximity, structural similarity, data alignment.

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080604, Database Management, 100%

Fields of Research (FoR) Classification

FoR code: 0806, Information Systems, 100%

Contents

List of Figures

List of Tables

Chapter 1

Prologue

1.1 Scope, motivation and objective

The World Wide Web was found in 1989 by Tim Berners-Lee ([12]). Since that time, it has been growing rapidly and providing a convenient environment for human to share information. By 2014, it is reported as having grown to approximately 2.9 billion users and the total of the order of one billion web sites ([2]). As such, this provides a massive repository of information and knowledge for mankind. However, majority of information on the web is represented in natural language or hypertext markup language (HTML). This can be easily read and understood by humans. However, since most of information on the web is in unstructured or semi-structured form, it is not easy to develop algorithms or automated processes to identify discrete entities, exact concepts from them, or identify relationships between the entities on the content of web pages.

A large number of research projects have been conducted to attempt to extract information from web resources ([17], [5], [41], [100], [9], [8], [114], [65], [109], [22]). The difficulty of information extraction is compounded by the fact that the quality of web pages varies from very high quality to rubbish or nonsense pages. In addition, there is an overlay of "noise" which arises from such sources as spelling errors, different formats, lack of standards, different abbreviations, language translation errors, etc. Information extraction (IE) is

intended to be an automatic extraction process to generate structured data from a collection of unstructured or semi-structured documents.

Unlike information retrieval, which is concerned with the return of relevant documents from a wide corpus of documents for a given query, information extraction systems attempt to generate structured information for post-processing, which is crucial to many applications involving data integration and search engine functions. Input to an IE process can be unstructured documents (like free text written in natural language) or semi-structured documents, such as web pages, which are pervasive on the internet. The result of the IE process is data in a structured form, which can be processed automatically by machines.

One of the typical tasks in information extraction is entity extraction. This has become an active and hot research topic over past decade. In 2009, Guo et al ([51]) reported that in search engine usage, entities occurs in about 71% of user queries. Whilst the providers of major search engines such as Google[1] or Bing[2] provide some information of how their search engines function, they do not provide details of engine structure, function and algorithm used. Nevertheless, it is well known that retrieval of information from queries by such search engines make considerable use of keywords and text matching techniques. There is little evidence to suggest that they have captured semantic information about objects, or relations between objects or entities on web pages. The process of entity extraction involves identifying entities, their attributes and relations between entities. This, therefore, is an indispensable task not only in providing answers to queries, but also generally in knowledge discovery from the web environment.

In the literature, several approaches have been proposed for extracting information relating to entities from the web. One of the popular approaches is to exploit a specific format of HTML web pages to build wrappers or templates based on manually labeled examples ([56], [99], [35], [37], [7], [96], [96], [89], [89]). However, this approach cannot scale up to the whole web because manual labelling is needed for the individual format of pages on each web site. Several attempts have been made to extract information from sentences by using

[1] http://www.google.com

[2] www.bing.com

natural language processing (NLP) techniques. In this approach, entities are required to be organised in sentences and their syntactic and grammatic features are exploited to build a statistical extraction model by utilising a training dataset with manually labelled data ([18], [79], [83], [87], [88]). Therefore, those approaches cannot be applied to extract information about entities which are not represented in sentences on web pages, such as lists, or tables, etc.

Meanwhile, a large number of web pages contain information about entities in the form of lists. Lists contain implicit records of entities and are also important sources of relational data about entities on the web. In such data sources, relationships between entities and their values are implicitly represented by the order of the values in the lists.

Figure 1.1 illustrates an example of a list of references on a web page in bibliographic domain. This list contains information about authors, paper titles, conference names and publication years. Clearly, if it was possible to extract information relating to entities in lists, implicit relations between the entities in the lists could be detected and give answers derived to related questions such as who published which papers, in which year, at which conferences as well as the co-authors of the papers. From the extraction results, structured tables could be built to present relationships between entities and discover knowledge in the domain.

Similarly, another example of the task in advertising domain can be seen in Figure 1.2. In this example, the list contains information about books. Each book has an author name, a book title, a publisher, a publication year, a serial number, and a price. Some books in the example miss information, such as Book No. 6 “Lost”, which does not include a publication year, or the book No. 10 “The Summer I turned Pretty”, where the serial number is not specified.

Such kinds of similar unstructured information are often available in several textual sources on the web, such as bibliographic information, postal addresses, advertisings, recipes, courses, conferences, research funds, etc. Nevertheless, the question of how to extract information of the data values from such unstructured lists is still a challenge. This is a practical and important research problem which has been frequently addressed in recent studies ([119],

1. Yi Yang, Fei Wu, Feiping Nie, Heng Tao Shen, Yueting Zhuang, and Alexander G. Hauptmann. "Web & Personal Image Annotation by Mining Label Correlation with Relaxed Visual Graph Embedding". IEEE Transactions on Image Processing (TIP), 2011.

2. Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. "Multiple Feature Hashing for Real-time Large Scale Near-duplicate Video Retrieval". In Proceedings of 19th ACM International Conference on Multimedia (ACM MM), 2011.

3. Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. "Transfer Tagging from Image to Video". Proceedings of 19thACM International Conference on Multimedia (ACM MM), 2011. (short)

4. Xiaofeng Zhu, Zi Huang, and Heng Tao Shen. "Video-to-Shot Tag Allocation by Weighted Sparse Group Lasso". Proceedings of 19th ACM International Conference on Multimedia (ACM MM), 2011. (short)

5. Jiajun Liu, Zi Huang, Heng Tao Shen, and Bin Cui. "Correlation-based Retrieval for Heavily Changed Near-duplicate Videos". ACM Transactions on Information Systems (TOIS), 2011.

6. Xiaoming Zhang, Zi Huang, Heng Tao Shen, Yang Yang, and Zhoujun Li. "Automatic Tagging by Exploring Tag Information Capability and Correlation". World Wide Web (WWW), 2011.

7. Yi Yang, Heng Tao shen, Feiping Nie, Rongrong Ji, and Xiaofang Zhou. "Non-negative Spectral Clustering with Discriminative Regularization". Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI), 2011.

FIGURE 1.1: An example of entity extraction from lists on bibliographic domain

1. Anderson, Laurie Halse. Wintergirls. Penguin/Viking Books. 2009. 9780670011100. $17.99.

2. Barnes, John. Tales of the Madman Underground. Penguin/Viking Books. 2009. 9780670060818. $18.99.

3. Booraem , Ellen. The Unnameables. Houghton Mifflin. 2008. 978 0152063689. $16.00.

4. Burg, Anne E. All the Broken Pieces. Scholastic. 2009. 978 0 545 08092 7. $16.99.

5. Collins, Suzanne. Catching Fire. Scholastic/ Scholastic Press. 2009. 978-0-439-02349-8. $17.99.

6. Davies, Jacqueline. Lost. Marshall Cavendish. 978 0 7614 5535 6. $16.99.

7. Garsee , Jeannine. Say the Word. Bloomsbury. 2009. 978 1 59990 333 0. $16.99.

8. George, Jessica Day. Princess of the Midnight Ball. Bloomsbury. 2009. 978 1 59990 322 4. $16.99.

9. Gill, David Macinnis. Soul Enchilada. Harper Collins/Harper Teen. 2009. 978 0 06 167301 6. $16.99.

10. Han, Jenny. The Summer I Turned Pretty. Simon & Schuster. 2009. $16.99.

FIGURE 1.2: An example of entity extraction from lists on advertisings

[94], [30], [32]).

In the literature, the problem of entity extraction from lists of field values is referred as the problem of information extraction by text segmentation (IETS) in which information of entities is organised in implicit semi-structured records on web pages ([94]). Solving the problem of IETS requires the tackling of several challenges, including data-entry errors, different formats, lack of standards, abbreviations, and large-scale data. There are several possible text segmentation schemes in information extraction to choose from list-specific wrappers or statistical segmentation models. However, since the field values in those implicit

semi-structured records are not machine-generated and they are represented in a textual representation in the implicit semi-structured forms, traditional wrapper-based methods ([35], [7]) which depend on HTML tags cannot be applied. A dominant approach in the literature to segment texts in input lists to extract field values is the deployment of statistical methods with two different techniques for generating training data. The first technique, which is called supervised approach, builds a training dataset manually ([95], [43], [16], [68], [85], [75]). Meanwhile, the second technique is called unsupervised approach, which exploits existing data in a knowledge base or reference tables to build extraction models automatically ([4], [75], [119], [30]).

The studies of Seymore et al ([95]) and Freitag et al ([43]) can be considered as the first research studies addressing this problem in the literature. In their work, a Hidden Markov Model (HMM) for recognising the field values in an input text was constructed from a provided training dataset. Later, this approach was extended in the system DATAMOLD ([16]). In the system, each state of an external HMM, which models the sequence of field values in an input text, contains an internal HMM. Each internal HMM is built as a model for recognising the value of each attribute. In their work, both internal and external HMM are trained from hand-labelled datasets.

After that, Conditional Random Fields (CRFs) ([68]) was proposed as an alternative model to HMM for information extraction. CRFs-based methods have been proven to outperform all previous learning-based methods in both theory and experimental evaluations for the problem of sequence labelling ([85], [75], [94]). They became popular in the field of information extraction because of their high flexibility and good extraction results. Currently, CRFs-based methods are statistical learning methods which are widely used in several information extraction systems ([94], [119]). They are more accurate and robust for extracting information about entities from such records because they can exploit an arbitrary number of rich and correlated properties of words in sentences.

In those statistical methods, an extraction model is trained using on a manually constructed training dataset which consists of a set of text segments and their labels. However, obtaining a large amount of training data, which includes the association between string

segments with their corresponding attributes, to build an extraction model requires a lot of laborious work and may be very expensive or even unfeasible in some situations. Although the quality of extraction results of HMM and CRFs are good, those supervised methods require the usage of a large amount of user-provided training data for the construction of extraction models.

Meanwhile, the general idea of the unsupervised approach is to exploit a pre-existing data source to alleviate the need for manually labelled training data when constructing a statistical extraction model ([4], [119], [30]). Agichtein et al ([4]) followed this idea and proposed an unsupervised method with HMM in their study. Later, Zhao and colleagues ([119]) proposed a similar technique but adapted the idea to CRFs. In those methods, a training dataset was built automatically by directly concatenating field values in a reference table or knowledge base. Since the training dataset was directly built from a knowledge base, those methods made a strong assumption about the overlap of format-related features between field values in the knowledge base and input lists. Due to this assumption, the statistical extraction models built from training datasets are able to capture features of field values in input texts and extract information. According to the experiments of Cortez et al ([30]), performance of the method in the study of Zhao et al ([119]) is quite low when testing data and referent table come from different sources. In addition, in term of time taken for execution, these methods have low performance because they execute an inference step to determine the order of field values and a training step to build training data and construct a new extraction model each time they perform an extraction ([119], [30]).

Cortez et al ([30]) have recently proposed ONDUX, an on-demand unsupervised approach for the problem of IETS. In their method, a knowledge base is employed to label text segments via some attribute matching functions of common terms between text segments in an input text and field values in the knowledge base. The labels are then used by a reinforcement phase to build a statistical extraction model to verify and potentially correct the labels which were generated by the matching phase. Therefore, a high overlap between a knowledge base and the source lists must be maintained in their work so that graphical extraction model can be generated correctly.

In general, existing statistical learning methods are flexible and robust but they require a large amount of manually labelled training data and have some limitations when applying the methods on web data in different domains. Obtaining such a labelling training dataset to be able to extract several types of entities which contain noisy tokens and style variations on the web may cost a lot of time and labor work. Meanwhile, although unsupervised methods can be utilised to build automatic extraction models to extract information of entities from lists, the methods mainly use the overlapping terms between field values in a knowledge base and input lists to build a statistical model for extracting the information of entities. They have not exploited the similarity of format-related features as well as approximately positional information of field values in different sequences inside a list to improve the quality of extraction process.

In order to extract information about entities from lists, our methodology is to exploit the overlap between a knowledge base and web data to annotate entities on documents on the web and build an extraction model automatically from rich labelling data. In our methodology, we exploit format similarity between a knowledge base and input lists and the structural similarity of sequences within a list when we build the extraction model. From the methodology, we firstly propose a self-supervised learning approach and define a general framework in which a knowledge base of a particular domain is exploited to build automatically a statistical extraction model to extract information about entities from lists on the web. The overlaps on format and structure between the knowledge base and an input list are exploited to build a training dataset for the extraction model. Examples of such a knowledge base can be reference tables or relations, ontologies, etc.

Building training data from a knowledge base requires some critical challenges to be addressed. Firstly, the field values or attribute values of entities in the knowledge base is typically “clean”, or stored in structured form, whereas input strings observed on the web lists may be “dirty”, contain a variety of noises, such as different formats, input errors, spelling errors, or using inconsistent abbreviations, etc. Therefore, the first challenge is how to annotate or label information of entities on web lists to build an extraction model from a knowledge base. In our approach, we identify and propose a dyadic representation of

semantic relations between a field value and an attribute of a concept by using its extensional and intensional representations. Specifically, an extensional relation between a field value and a concept or a label is defined by the similarity between the content of the field value and all members of the concept or the label within a knowledge base. Alternatively, the similarity between a field value and a concept can be expressed in an intensional definition by specifying a set of lexical-syntactic patterns to identify the field value. We design those patterns in a set of rules for some primitive datatypes to check the membership relation. Due to this labelling technique, we can obtain high performance on segmentation and matching process which helps to improve the overall performance of extraction phase in our proposed framework.

Moreover, whilst a sequential model is used to capture the transitions between labels in sequences of an input list, positional information of text segments in sequences also play important role when assigning labels to the text segments. Intuitively, the labels of the same concepts in different sequences of an input list are often located in similar positions. However, the labels of text segments, which are assigned in a matching step, may not always be in fixed positions in the sequences of an input list. The sequences of an input list may contain different numbers of text segments. Moreover, the number of field values and their lengths may also be different in different sequences. In other words, positional information of labels is approximate in different sequences of a list. To capture positional information of labels in a list, we propose a novel proximity-based positional model for labels to combine with a sequential model, which captures the transitions of labels in different sequences of a list, to build an information extraction model for entities from the list. Our proximity-based positional model can capture the distribution of labels in different positions in an input list when we revise labels of text segments. Due to this model, we can improve the quality of information extraction in our framework.

Eventually, the process of building a statistical extraction model from a knowledge base mainly exploit the overlap between the knowledge base and web lists to assign labels for text segments. In order to reduce the dependence on the overlap between a knowledge base and an input list, we exploit the internal structural similarity between text segments in the

sequences of the list to align text segments into groups before we assign labels for them. We firstly propose a novel structural similarity between text segments, and then devise a data shifting-alignment technique to align and cluster text segments into groups before their labels are revised by a graphical model.

1.2 Summary of major contributions

Following are the major contributions of this thesis to information extraction area.

1. We define a general self-supervised learning framework for building a statistical extraction model to extract information about entities from textual lists. In our proposed framework, a knowledge base is exploited to assign labels for text segments and build automatically a statistical model for extracting information about entities from lists.

2. We identify and propose a dyadic representation of semantic relations between a field value and an attribute or a label by using its extensional and intensional representations. Then we exploit the representations to conduct a format-enhanced labelling technique to improve the quality of labelling step in our framework and overall extraction process.

3. A proximity-based positional model for labels is proposed in our study to capture approximately positional information of labels annotated in different sequences of an input list. According to our experiments, our proposed model can help to improve the effect of extraction model on entity extraction from lists.

4. To reduce the dependency on shared terms between a knowledge base and an input list when building an extraction model, we propose a novel structural similarity measure between text segments and devise a data shifting-alignment technique to cluster similar segments into groups before their labels are revised by a graphical model. According to our experiments, our proposed method requires less overlapping terms than the current state-of-the-art method whilst it still keeps high performance on the quality of information extraction from textual lists.

1.3 Structure overview

After chapter 1, the remaining chapters of this thesis are to present in detail our ideas and technical solutions to attain the above objectives. The chapters are organised as follows.

Chapter 2 provides the background of this thesis on entity extraction from web data. Sections 2.1 and 2.2 correspondingly introduce the problem of entity extraction and entity extraction by text segmentation on web documents as well as their related studies in the literature. Section 2.3 surveys related work on approximate matching techniques for entity names on web documents. The majority of the content of this chapter is partially reported in our article in [58].

Chapter 3 presents our self-supervised learning framework for entity extraction from textual lists as well as our proposed format-enhanced labelling technique. In section 3.2, we formally defines the problem entity extraction by text segmentation and describes the self-supervised learning framework. After that, section 3.3 presents our proposed format-enhanced labelling technique. The experiments and comparisons are illustrated in section 3.4. The ideas and results of this chapter are partially published in [59] and [61].

Chapter 4 presents a proximity-based positional model for labels to capture approximately positional information of labels in an input list to improve the quality of the extraction model in our proposed self-supervised learning framework. Firstly, the proximity-based positional model for labels is developed in section 4.2. Afterward, section 4.3 illustrates our experimental results to prove the efficiency of the model in entity extraction from textual lists. This chapter is a refinement of our publication in [61].

Chapter 5 presents a novel structural similarity measure and a data shifting-alignment technique to cluster similar text segments into groups or columns before the labels of text segments are revised by a refinement phase in the self-supervised learning framework. Firstly, a structural similarity model for text segments is defined in section 5.2. Then we explain our proposed data-alignment technique and algorithms in section 5.3. Section 5.4 illustrates experimental results and provides a discussion on the results. The ideas and results of this chapter are partially published in our papers in [60] and [62].

Chapter 6 summarises the content of the thesis and proposes some suggestions for future studies based on the current research results.

Chapter 2

Background

2.1 Introduction

Information extraction (IE) is an automatic extraction process to generate structured data from a collection of unstructured or semi-structured documents. Unlike information retrieval (IR), which is concerned with how to return relevant documents in a corpus for a given query, information extraction systems generate structured information for post-processing, which is crucial to many applications of data integration and search engines. The input of the IE process can be unstructured documents like free text written in natural language or a semi-structured documents, which are pervasive on the internet. The result of the IE process is data in a structured form, which can be processed automatically by machines.

The extraction of structured data from noise and unstructured sources is a challenging task that devoted several efforts of scientists in the area of information extraction. One of typical tasks in information extraction is entity extraction, which has become an active and hot research topic over the past decade. According to the study and analysis of Guo et al ([51]) on search engine usage, named entities occurs in about 71% of search queries of users. Nevertheless, current search engines such as Google or Bing, which support users to search information on the web according to their queries, are mainly based on keyword or text matching techniques and have not captured semantic relations between objects or

FIGURE 2.1: A query about entity on Google search engine

entities on web documents. Figure 2.1 illustrates an example in which users want to find the location where SIGMOD conference organised in 2013. Google search engine returns web documents containing the terms in the query, which may not be what users actually need. Therefore, the problem of identifying named entities, their attributes as well as the relations between entities is an indispensable task not only in query answering but also in knowledge discovery from the web environment.

In this chapter, we provide a background on entity extraction, which is an important task in information extraction, and introduce some methods for string matching on names of entities in the literature. In the scope of this literature review on entity extraction, we focus on

the methods of entity extraction and provides an insight into the challenges in this problem. Firstly, we introduce the approaches in the history and current research which were used to tackle two typical sub-tasks in entity extraction, including entity recognition and relation extraction of entities on documents or web pages. The first sub-task mainly regards how to detect interested named entities on documents whilst the second one is to extract information of some specific entities, such as their attribute values as well as relations between entities.

In the remaining sections of this chapter, we present the background on entity extraction and entity extraction of text segmentation as well as related studies for remaining chapters of this thesis. Moreover, related studies on name matching will be also represented in this chapter. Section 2.2 and 2.3 respectively present an overview on entity recognition and relation extraction between entities on documents or web pages. After that, section 2.4 focuses on the problem of entity extraction by text segmentation and their shortcomings. Next, section 2.5 surveys previous studies on string matching techniques between entity names. Finally, section 2.6 presents concluding remarks of this chapter.

2.2 Named Entity Recognition

The problem of named entity recognition (NER) was originally defined at the Message Understanding Conference 6 (MUC-6) ([49]) in 1996. The goal of the task is to identify names and types of entities on text documents. They can be the names of people, organisations, geographic locations, times, currencies, and percentage expressions. NER problem has been extensively studied in the literature. In general, most previous approaches can be categorised into two categories, including rule-based and statistical approaches.

2.2.1 Rule-based approach

In order to recognise types of entities on text documents, rule-based approaches define heuristic rules to identify named entities within documents in a particular domain. These

rules are manually built by experts in the domain to extract information of entities. In general, each rule in a rule-based system has two components, comprising of context patterns of the rule and action of the rule. The context patterns defined the pre-conditions of a rule so that the rule can be applied to perform a corresponding action. These pre-conditions capture various properties and the contexts that entities appear in a document. Most of rule-based systems are mainly based on regular grammars, which can be parsed by a deterministic finite state automata (DFA) ([57]), to define the patterns. The action of a rule is used to assign a label or an entity type for one or a sequence of tokens in text.

In rule-based approaches for NER problem, most rule-based systems regard the representation of rules so that they obtain the efficiency in matching processes and application of rules for extraction. Therefore, several rule representation formats have been defined and evolved over the years, e.g. the common pattern specification language in FASTUS ([56]), regular expression in WHISK ([99]), JAPE language in GATE ([37]), XLog or Datalog expressions in DBLife ([96]), and algebraic language in Avatar ([89]).

In FASTUS system ([56]), a set of rules are manually encoded in patterns to extract information from sentences in input texts. For example, the pattern "<City> is a capital city of <State>" can be used to match with sentence "Brisbane is the capital city of Queensland" to extract information of the city "Brisbane" and the state "Queensland" in the sentence. The patterns could be defined more complex to capture the content encoded in relative clauses. Similarly, WHISK exploits regular expression to encode rules. For example, a rule of WHISK in Figure 2.2 says that if an advertising string starts with a digit followed by the word "bedroom" and a then a number, then the digit will be returned as the number of bedrooms and the number will be the prices of the rental.

Whilst Cunningham and colleagues ([37]) exploited some syntax of Java programming language to define rules in JAPE language, Shen et al ([96]) defined extraction rules as predicates in Prolog. Figures 2.3 and 2.4 illustrates the examples of a rule to recognise person names in JAPE ([37]) and XLog ([96]), respectively. The rules in the examples says that a string for a candidate person name is recognised if it can be matched in a dictionary of first names and it contains an initial capital letter. Similarly, Reiss et al ([89]) proposed an

```
ID:: 1
Pattern:: *(Digit) 'BR' * '$'(Number)
Output:: Rental | {Bedroom $1} {Price $2}
```

FIGURE 2.2: An example of a rule in WHISK

```
Rule: CandidatePersonName
Priority: 1
(
{ Lookup.kind == firstName }
{ Token.orthography == initialCaps }
):match
–> :match.kind = "CandidateName";
```

FIGURE 2.3: An example of a rule in Jape

algebra that includes text-specific operations and exploit properties of the new operators to perform some optimisation techniques.

In general, in order to perform a rule matching process, tokens along with their features in a document are firstly extracted. The features of a token can be the representation string of that token, its part-of-speech, list of dictionaries in which the token appears, orthography type of the token such as upper-case word, lower-case word, mixed case word, number, special symbol, space, punctuation, and so on. These tokens and their features are then used to match

```
CandidatePersonName(d, f, l) :-
docs(d),
firstNamesDict(fn),
match(d, fn, f),
match(d, "[A-Z][a-z]+", l),
immBefore(f, l);
```

FIGURE 2.4: An example of a rule in XLog

with the pre-conditions of rules to find applicable rules. Eventually, those matched rules are applied to perform corresponding actions to determine the types of entities. According to the report in [37], the default extraction module ANNIE built on GATE system produces about 85% precision and recall on news domain.

In most rule-based systems, dictionary matching techniques are crucial tasks in performing many basic extraction tasks such as identifying person salutations (e.g., "Mr", "Ms", "Dr") or identifying occurrences of known first names, nouns or phrases. Several studies in database community have recently addressed the problem of approximate entity extraction from a dictionary to improve the efficiency of information extraction tasks ([24], [25], [6], [23], [26], [72], [111], [70]). The goal of the problem is to identify all substrings in a document which match approximately with a set of predefined entity names in a dictionary. Most methods focused on novel filtering techniques to eliminate as many candidate substrings as possible to gain efficiency in matching process. In general, existing approaches for the problem of approximate entity extraction from a dictionary can be divided into three categories, including inverted-index-based, signature-based, and hybrid-based approach.

Inverted-index based approach ([24]) tries to index all distinct tokens in the dictionary and builds record ids for strings in the dictionary. Given a query substring, this approach merges the record ids of tokens in the query and directly identifies the entity in the dictionary which best matches with the query. Meanwhile, the signature based approach computes a set of signatures of the query substring and dictionary strings before they are matched together. The underlining idea of this approach is that a query substring is similar to a dictionary string if their sets of signatures are overlapping. The signature of a string represents important information of an entity name and it can be defined by low frequent tokens ([24], [25], [26]) or subsets of tokens ([6]) in the string.

Chakrabarti and colleagues ([23]) proposed a hybrid approach which combines the benefit of inverted-index-based and signature-based approach. In their work, an *inverted signature-based hashtable* (ISH) is employed to store a list of signatures per indexed token. As compared to inverted-index based methods, their approach replaces each record identifier with the set of signatures of the string corresponding to the record identifier. The experiments in

their work showed that the method outperformed previous methods in term of running time.

A similar idea of employing the signature of a string for matching entities can be found in the study of Chaudhuri et al ([26]). In the study, the authors proposed to expand a reference dictionary of entities by mining large document collections. Then the input string can be matched exactly against entities in the expanded reference table. Similarly, Lu at al ([72]) proposed signature-based inverted lists combined to improve the study of Chakrabarti et al ([23]) by using a tighter threshold. However, a limitation of these studies is that they only consider token-based similarity measures. Whilst entities in the dictionary are usually cleaned and standardised, they are often represented in "dirty" form in documents, such as typographical or orthographical errors. Therefore, it may miss some matches if only token-based similarity for matching entities is employed.

Motivated by those drawbacks, Wang et al ([111]) proposed NGPP, a neighbourhood generation-based method for matching entities with edit-distance thresholds. In their work, two input strings are firstly split into multiple partitions. The two strings are similar only if there exists two partitions of the two strings having edit distance less than or equal to one. The neighbourhoods of partitions are generated by deleting one character from the partitions. If two partitions share a common neighbour string, their edit distance will not be greater than one. One of drawbacks of this method is the requirement of larger index sizes as compared to q-gram-based method ([70]) because all variants of partitions need to be indexed.

Recently, Li and colleagues in [70] has devised Faerie, an efficient filtering method for counting the common tokens between a candidate substring and an entity in dictionary-based entity extraction. In their work, two entities are similar if the size of the set of common tokens/q-grams is not smaller than a threshold. They exploited heap-based filtering algorithms to utilise shared computation across the overlaps in the substrings of the document. Due to avoiding unnecessary redundant computation, they can obtain effective performance in running time when compared to the studies of Chakrabarti et al ([23]) and Wang et al ([111]).

In general, one of the advantages of rule-based approach for the problem entity recognition is that the execution time of rule-based systems is shorter than methods in other approaches ([96], [89]). In addition, rule developers can easily control rules to obtain certain optimisation for some specific domains, such as the extraction of phone numbers, zip codes, dates, and time. However, this approach requires domain experts to define manually rules for information extraction, and those rules can be rigid and not general enough to cover all cases of real data. This may directly affect the completeness of entity types in the results of rule-based systems.

2.2.2 Statistical approach

The underlining idea of the statistical approach for NER is to convert the problem of entity recognition into two other problems, including the problem of decomposition of unstructured texts and the problem of labelling the parts of decomposition. The parts of decomposition are commonly represented in one of two prevalent forms: tokens and word chunks.

In the representation of token form, an unstructured text is simply split into a sequence of tokens according to a set of delimiters, e.g punctuation marks. The next step is to assign an entity label to each token in that text. After that, entities in a document can be marked as a consecutive tokens with the same entity label. In order to label tokens in a text, a large number of extraction models have been proposed in the literature. The basic idea of those models is to determine the label of a token by exploiting the features of that token and its neighbour tokens in text. Those features can be defined in several ways according to particular domains. They can be the representation string of a token, punctuation marks, or the occurrence of words in a dictionary, etc.

In the case of the representation of work chunks, some techniques in natural language processing (NLP) ([73]) may be applied to identify noun chunks in a sentence. The features of each chunk can be defined by the combination of the features of tokens in that chunk. This allows to define more robust features which may help to identify entities because the properties of all tokens in each chunk can be captured. Afterward, the labelling process for

the chunks in text is similar to the labelling process for tokens. However, instead of assigning labels to tokens, ones assign labels to word chunks.

In the labelling phase, a model is firstly trained from a training dataset. Then it is used to identify information of entities from unstructured texts. One of the ways to assign labels for tokens is to view the problem of token labelling as the problem of classification in which the model must determine whether a token is assigned a particular label or not. Therefore, any existing classifiers can be used to classify tokens. The study in [53] is an example of research work that used a Support Vector Machine (SVM) to extract meta-data of citations.

Nevertheless, the labels of adjacent tokens are seldom independent of each other and they can be used to determine the label of a token. Consequently, different models were proposed to capture the dependency between the labels of adjacent words, such as Hidden Markov Models (HMMs) ([13]), Maximum entropy Markov models (MEMM) ([76]), and Conditional Random Fields (CRFs) ([68]). Currently, CRFs based methods are the current state-of-the-art machine learning technique and outperform all previous machine learning based methods in both theory and experimental evaluations for the problem of sequence labelling in machine learning based approach ([85], [94]). Moreover, the implementations of CRFs such as CRF++ ([104]), Mallet ([77])), CRF Project ([93]) are already available on the internet. Users can re-use these tools to define sets of necessary features to extract information in their particular domains.

Figure 2.5 demonstrates an example of a training dataset for a sequence on bibliographic domain for Mallet library ([77]. The dataset contains a set of rows and each row includes a token or a chunk, its features and assigned labels. As can be seen in the figure, the token "Kimon" has three features: BIBTEX_AUTHOR, INICAP, KIMON. These features are string constants predefined by developers when they build the system. By that way, the number of features to build the CRFs models can be justified and controlled easily in the system. Moreover, the prefix "B" in the labels are used to distinguish the staring position of a new label in a sequence.

Like other machine learning methods, the training dataset plays an important role in training an extraction model. The performance of training models are directly affected by

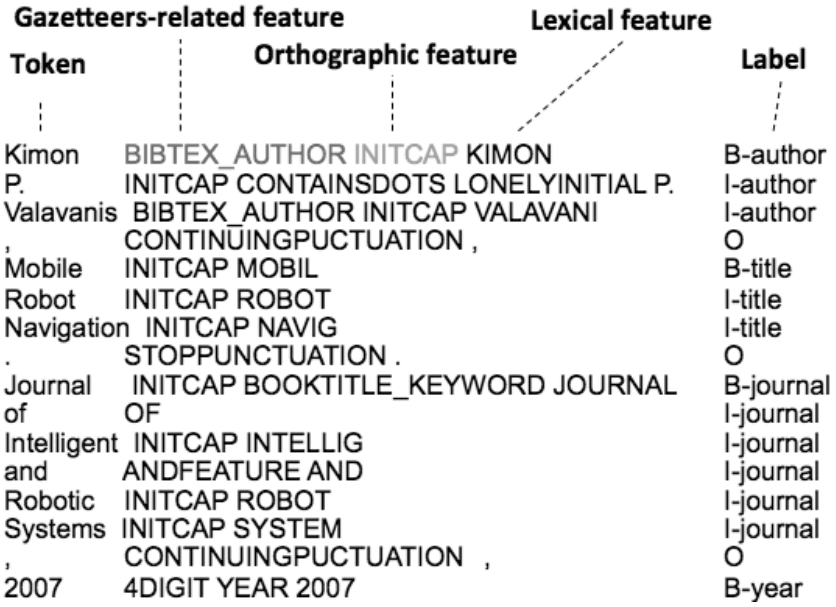


FIGURE 2.5: An example of training data for CRFs methods

the chosen features as well as quantity and quality of training data. Therefore, the learning methods in this approach requires a lot of time and laborious work to manually build such a training dataset. This limits the usage of those methods in web-scale applications.

2.3 Relation Extraction

Relation extraction is one of important sub-tasks of entity extraction. Different from the problem of entity recognition, which focus on how to identify entity and label the types of entities in textual documents, the goal of relation extraction task is to identify and extract semantic relations between entities within text. It is motivated by the requirements of extracting and structuring the attributes of entities from documents so that machines can process automatically. For example, the sentence ”Microsoft bought Skype for $8.5 billion” can be approximately represented by the relation “ACQUIRE(Microsoft, Skype)” which described the acquisition relationship between two entities “Microsoft” and “Skype”. Most previous studies assume that the entities in document have been identified by some simple matching techniques before they perform relation extraction between them.

The problem of relation extraction has been a hot and active research topic for recent years and several research studies have been proposed to extract relationships between entities on document or web pages. In those studies, each relation has a type signature that decides the entity types in the arguments of that relation. For example, the relation birthPlace $\subseteq$ Person $\times$ Location, graduatedAt $\subseteq$ Person $\times$ Univerisity are binary relations. Most previous studies concentrated on the extracton of binary relations from documents ([5], [41], [65], [9], [22]). Recently, some others look to the extraction of higher-arity relations or records from web ([20], [19], [52], [50], [71], [108]).

In general, previous research models relation extraction problems in one of the two following scenarios. The first one is to design algorithms that identify the type of relationship between given entity pairs in a document. The second scenario in relation extraction is to retrieve all entities that satisfy a given relationship type. Based on the scenario, the relation extraction problem in their studies is modelled on one of the following approaches: a supervised approach, semi-supervised approach, or an unsupervised approach is employed to extract relations between entities on documents.

2.3.1 Supervised approach

Supervised learning approach is a popular technique for identifying a relation type between any given entity pairs in sentences or textual documents. This approach assumes that two entities in a relation in a document are in close proximity to each other and occur in the same sentence. Therefore, relation extraction problems from natural language text can be formulated as the problem of identifying whether there are any relationships between two particular entities in a sentence or not.

Given a pair of entities in a sentence, in order to determine which relation is mentioned in the sentence, the supervised learning approach exploits techniques in natural language processing (NLP) to parse the sentence into well-defined structures. Due to the parsing process, various syntactic and semantic properties in the well-defined structures of a sentence are employed as a set of features for the detection of relations between the entities.

Formally, let's consider a sentence $S = w_1 w_2 ... e_1 ... w_i ... e_2 ... w_{n-1} w_n$, where e_1 and e_2 are entities and w_i's are words in the sentence S. The goal is to determine which relation is be represented by words w_i's in S. To solve this problem, supervised learning approach employs machine learning techniques to study a binary classification function f to check whether the two entities e_1 and e_2 are related by a relation r or not. The classification function f is frequently performed by using a real-valued function $f : O \rightarrow R$ where each $o \in O$ represents a set of features extracted from entities e_1 and e_2 in the sentence S and o is assigned to positive class if $f(o) \geq 0$, and to negative class otherwise.

In order to extract the features of a relation, supervised learning approach exploit NLP techniques to parse a sentence S and obtain a structured representation to define o and the binary classification function. In other words, the problem of relation extraction is formulated as a classification problem and relations of unseen entity pairs are classified by a target function which is learnt from a manually labeled training dataset. Previous study applied Support Vector Machine (SVM) ([18], [79], [83], [87], [88]) to learn the target function from training data and used it to determine the label of the relation between novel entity pairs in the sentence.

Given a vector space and a set of training points, i.e. positive and negative examples, the goal of SVM is to find a separating plan $f(\overrightarrow{x}) = \overrightarrow{w} \times \overrightarrow{x} + b$ where $w \in R^n$ and $b \in R$ to separate examples in training data. Both parameters w and b are learned from labeled training data by Structural Risk Minimization principle ([107]) in statistical learning theory.

One of strong point of SVM is the possibility of applying a kernel method ([80]) to implicitly map data into a new space where examples in training data are easy to separate by a plan. In that way, each object o to be classified is mapped into a feature space $\overrightarrow{x}$ via a feature function $\phi : O \rightarrow R^n$, where O is a set of objects for classification. From that definition, the decision hyperplan $f(\overrightarrow{x})$ in SVM is accordingly rewritten as in equation 2.1, 2.2, 2.3, and 2.4. We note that the weight vector $\overrightarrow{w}$ in equation 2.1 has been replaced by a function of $\overrightarrow{x_i}$ with the new parameters α_i.

$$f(\overrightarrow{x}) = (\sum_{i=1..l} y_i \alpha_i \overrightarrow{x_i}) \times \overrightarrow{x} + b \tag{2.1}$$

$$f(\overrightarrow{x}) = \sum_{i=1..l} y_i \alpha_i \overrightarrow{x_i} \times \overrightarrow{x} + b \tag{2.2}$$

$$f(\overrightarrow{x}) = \sum_{i=1..l} y_i \alpha_i \phi(o_i) \times \phi(o) + b \tag{2.3}$$

$$f(\overrightarrow{x}) = \sum_{i=1..l} y_i \alpha_i K(o_i, o) + b \tag{2.4}$$

where y_i is equal to 1 for positive and -1 for negative examples, $\alpha_i \in R$ with $\alpha_i > 0$, o_i is an instance in training data , l is the size of training data, $K(o_i, o) = \phi(o_i) \times \phi(o)$ is the kernel function associated with the mapping ϕ.

A kernel function $K(o_i, o)$ is a scalar product on a possibly unknown feature space. Therefore, it is not necessary to apply the mapping function ϕ for each object. Instead, the kernel function $K(o_i, o)$ can be used directly and it can be defined based on the structured presentation $T(S)$ of a sentence. The kernel function can be considered as a similarity measure between the structures that describe entity pairs in a particular relationship.

A kernel function can be defined by combining basic kernel functions with additive or multiplicative operators. Each basic kernel measures the similarity between pairs of parsed trees by counting the number of overlapping fragments between them. According to the study of Moschitti et al ([79]), three important characterisations of a fragment type are sub-trees, sub-set trees, and partial trees. Formally, a kernel function between two trees T_1 and T_2 can be defined as in the equation 2.5.

$$K(T_1, T2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \tag{2.5}$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, Δ is a function returning the number of common fragments rooted in nodes n_1 and n_2.

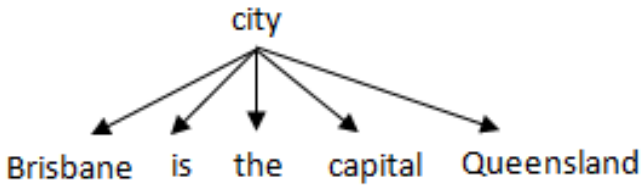


FIGURE 2.6: An example of dependency tree based on words.

Different parsed trees of a sentence can be exploited to define a kernel function. In general, two common structured representations, which are often used in the literature, are dependency tree and constituent tree. A dependency tree of a sentence encodes grammatical relations between words in a sentence. In a dependency tree, words are placed in the nodes whilst dependency types of those words are represented by edges in the tree. Except the root node, every node in a dependency tree has exactly one parent. An example of dependency tree based on words to represent the sentence "Brisbane is the capital city of Queensland" can be seen in Figure 2.6.

Culotta and colleagues ([36]) proposed the usages of dependence tree to represent the grammatical dependencies in a sentence. This work was based on the hypothesis that instances containing similar relations will share similar sub-structures in their dependency trees. In this work, a tree kernel function over dependency trees was defined and incorporated this kernel within an SVM to extract relations from documents. Bunescu et al in [18] proposed a similar method as in the study of Culotta et al ([36]) but they specified that the shortest path between the two entities in dependency tree, which is a smaller representation as compared to previous work, can improve the performance of relation extraction system.

Meanwhile, a constituent syntactic parse tree was proposed by Zhang et al ([118]) to represent the syntax of a sentence for relation extraction. Constituent tree is another way to represent and encode structural properties of a sentence. This kind of tree contains the constituents of a sentence such as noun phrases (NP), verb phrases (VP), and part of speech (POS) tags. Figure 2.7 is an example of the constituent parse tree for the sentence "Brisbane is the capital city of Queensland".

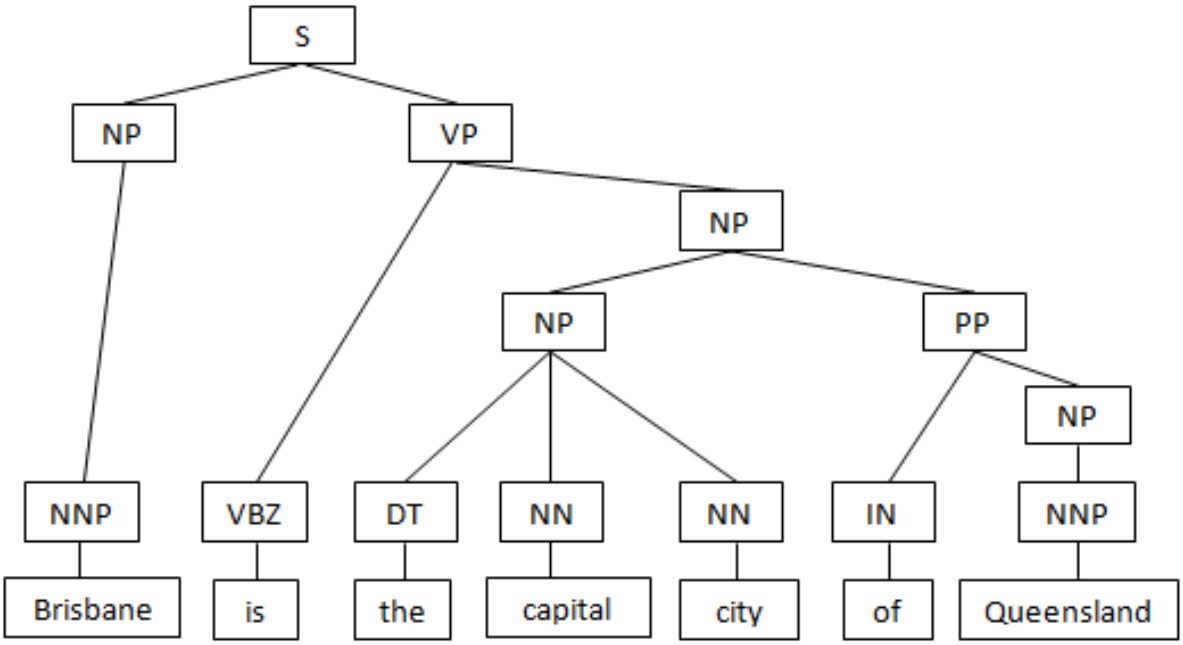


FIGURE 2.7: An example of constituent tree.

Moreover, Moschitti et al in [79] suggested the combination of constituent and dependency parse trees to represent the syntax of a sentence for relation extraction. This idea was continued to improve by augmenting the semantics concerning to entity types and lexical sequences ([83]). Recently, Reichartz et al in [88] have proposed a novel method of using typed dependency parse trees which include labeled edges and nodes. The labels and topology of a typed dependency parse tree contains semantic clues which were exploited in this work for relation extraction. The empirical evaluations showed that it is the current state-of-the-art method as compared to previous kernel-based methods.

In general, this learning approach for relation extraction requires expensive deep linguistic parsing techniques in NLP. Moreover, a training dataset of sentences with correct labels must be manually built to train a model which is then used to predict new semantic relations. This makes supervised methods difficult to extend to several types of relations and apply to large-scale applications for extracting information from the web. In addition, this approach requires POS tagging and sentence parsing which may consume resources and be prone to errors. Syntactical parsing may deal with the problem of syntactic ambiguity when there are

more than one syntax tree representing for a sentence. Therefore, in spite of several extensive researches in NLP community, the current accuracy values of this approach for relation extraction still range in the neighbourhood of 50%–75%, even in the ACE standard benchmark dataset ([88]).

2.3.2 Semi-supervised approach

An alternative scenario of extracting relationships between entities is to extract all entity pairs of one or more given relationships occurring in a document or a corpus. An obvious idea to solve this problem is to exploit the representation of those relations on documents and define patterns encoded into rules to extract information of entities. Therefore, this approach is also called as pattern-based approach.

The usage of patterns for information extraction from natural language documents has a long history. In 1992, Hearst et.al ([54]) proposed the usage of syntactic patterns to identify instances of predefined relationship types from free text. These patterns are defined by using regular expressions on Part of Speech (POS) of words in sentences. For example, the patterns of the form "<Noun> such as <List of Noun Phrases>" or "<Noun> including <List of Noun Phrases>" was used in [54] to extract *instanceOf* relations, which is also called hypernym relations. The patterns in the example indicate that the noun phrases in the lists are the instances of the entity type specified in the noun of the patterns. When the patterns are applied on a document which contains the text "companies such as Sony, Amazon, Apple, Microsoft", the relations between the entities "Sony", "Amazon", "Apple", "Microsoft" and the entity type "companies" can be extracted. By this way, NER problem can be considered as a special problem of relation extraction in which only *instanceOf* relation between entities and their types is extracted. Similarly, Berland et al ([11]) also employed similar Hearst patterns for extracting *partOf* relation in a very large corpora.

Although Hearst's patterns can yield relatively high precision, the patterns which are manually defined become difficult to adapt so as to deal with arbitrary target relations. Therefore, Brin ([17]) suggested a new method, called DIPRE, which exploited the duality of facts

and patterns in documents and applied to method to extract relation instances between authors and books.

In Brin's idea, each relation R is described by specifying the types of the entity pairs which form the arguments of the relation R. An initial set of seed facts for one or more relations is firstly utilised to find automatically the markup, textual, or linguistic patterns of relations from a corpus. Due to the process, general patterns are generated and then applied to identify new fact candidates from the corpus. [90] is a similar study which also employs this method to extract information in the terrorism domain. The way of studying such patterns from a set of seed entities is also called bootstrapped method or semi-supervised learning method in the literature.

Several large-scale projects have improved, enriched and deployed those patterns in their systems. Snowball ([5]) is one of such projects which employed the pattern-entity duality of Brin ([17]) to generate extraction patterns in the system. These patterns are used to extract new tuples and augment into Snowball as new knowledge. The evaluation measure for patterns and tuples was defined in the study to rank and choose the generated patterns and new seed tuples. However, the extraction patterns in Snowball are mainly based on strict keyword-matching. Therefore, although it can identify highly accurate results, the recall of this method will be limited. In addition, Snowball only focuses on extracting a specific type relation, e.g relation between companies and headquarters. The measures to evaluate generated patterns and tuples cannot be adapted to any types of relation. Those limitations motivated a novel system, called StatSnowball ([120]) which applied a statistical model to select good patterns and relation tuples. StatSnowball can perform both traditional relation extraction like Snowball to extract pre-specified relations and open information extraction (Open IE) ([9]) to identify any general types of relations.

KnowITAll ([41], [42]) is another project that exploits web search engines to build patterns for information extraction. It used patterns to extract from large-scale web pages with 54,753 facts regarding cities, states, countries, actors, and films. The patterns in KnowITAll was manually built and partially adapted from Hearst's patterns ([54]). The main difference between KnowITAll and other bootstrapped methods is that KnowITAll does not require

any manually tagged training sentences or initially entity seed set to build patterns. Instead, KnowITAll employed a set of initially generic extraction templates, which is domain-independent, to induce a set of seed instances.

Following is an example of the generic template in KnowITAll: “NP1 such as NPList2”. The template in the example indicates that each simple noun phrase (NP) in NPList2 is an instance of the class named in NP1. This template can be used to extract the company names from such sentences as “Companies such as Sony, Amazon, Apple Inc., Microsoft used the power of digital rights management”.

The patterns in KnowITAll are used to formulate the queries to retrieve web pages from search engines and extract appropriate sentences in downloaded web pages. Those queries are generated by the target classes combined with a query expansion technique to improve the quality of searching results. Finally, a measure based on web-scale statistics which is computed by the number of hits returned in response to a query was defined to assess the probability of an extraction generated by the system.

The approach of KnowItAll requires predefined domain-independent rules for relations and a large numbers of search engine queries as well as webpage downloads. Therefore, Banko and colleagues proposed TextRunner ([9], [117]) which is an improvement of KnowITAll in term of precision. Instead of taking relation names as input, TextRunner automatically discovers possible relations of interest from a small corpus during processing. It deploys a deep linguistic parser to analyse several sentences from a corpus sample to obtain their dependency graph representations. Then the syntactic structures which are shared by any two base noun phrases in the parsed sentences are used to train a Naive Bayes classifier. This classifier will be utilised to label candidate tuples as “trustworthy” or not. The strings between entity pairs are normalised and used as the relations between entity pairs. The learner of TextRunner supervises their learning process automatically by labelling its own training data.

Although TextRunner project pursues an ambitious goal of extracting all instances of all meaningful relations from Web pages, it has two shortcomings in its approach. Firstly, the relation extracted by the system are not well-defined relations because they are word phrases

between entities. Secondly, the entities in the knowledge base may be ambiguous because different entities may have the same name or an entity may have different aliases. A similar work described in LEILA project ([100]) also exploited linguistic structures of sentences and machine learning techniques to generalize robust patterns for relation extraction.

Recently, a number of projects have applied rule-based approach with specific focus on information extraction on Wikipedia. They exploited the semi-structured parts such as infoboxes and category system of Wikipedia to design patterns and rules to extract facts and construct ontologies of entities ([8], [65], [102], [114], [112], [115], [116]). DBPedia ([8]) pioneered the massive extraction of infobox facts from Wikipedia. It used recall-oriented techniques to extract all attribute-value pairs in infoboxes into its knowledge base. Nevertheless, the extracted values of attributes do not necessarily correspond to known entity types. As a result, the values and attribute names can have mixed quality because of non-unique format and naming.

YAGO-NAGA ([65]), on the other hand, is another project that also proposed a rule-based IE system to build a huge knowledge base from Wikipedia. As compared to DBPedia, YAGO-NAGA did not attempt to harvest all attributes and values to avoid the diversity of their aliases and noise. Instead, the frequent attributes in infoboxes in Wikipedia are extracted by using a suit of rules and their corresponding values are normalised before they are stored in YAGO ontology. In YAGO-NAGA, the structured template of infoboxes and category system of Wikipedia was exploited to extract facts, which were then combined with the taxonomic class system of WordNet thesaurus, to build the ontology of entities. SOFIE ([102], [81]) is an extension of YAGO in which it incorporates a reasoning process to check the valid of extracted facts.

Kylin project ([114], [112], [115], [116]) is a recent study to extract information of infoboxes by combining with machine learning techniques for extraction. Its system goes beyond DBpedia and YAGO by extracting information not only from the infoboxes and categories, but also from the content of articles on Wikipedia. In order to extract information from articles, Kylin considered information in existing infoboxes as a source of training data to build a learning model for extraction.

In general, an assumption in the bootstrapped method is that any given entity pairs in an initial seed set cannot participate in more than one relationship with each other. This may not be difficult to obtain in practice because not all sentences containing an entity pair support the relationship type. For example, in the two following sentences: "Tom is a friend of Jack" and "Tom is a colleague of Jack", The Tom and Jack entities participate in two different relationship types. In addition, this approach requires that all entities have been marked in a document. This may lead to some difficulties when entities in the initial seed set occur in different aliases in text. Moreover, bootstrapped based methods demand a good evaluation measure and strategy to assess and control the quality of generated patterns.

The relation extraction methods mentioned above studied the lexical-syntactic patterns from natural-language inputs. Several studies also exploited the structure of web sources to define wrappers or rules to extract information of entities. They constructed the wrappers for fact extraction from HTML headings, tables, lists, form fields, and other semi-structured elements within a web site. Some tools were implemented to simplify the process of defining wrappers. For example, W4F toolkit ([91]) defined a language that allows users to describe a wrapper and map extracted data into XML, Lixto ([10], [45]) provided a fully visual and interactive user interface for wrapper specification process. Meanwhile, some other works tried to develop methods for learning structures from examples by comparing HTML pages and automatically generating a wrapper based on the similarities and differences of those web pages ([67], [7], [35], and [34]). Although wrappers provide an effective mechanism to extract information and can be learned by using a small number of examples, they only work well on particular sites for which they define. Therefore, this limits their usage for web-scale extraction.

SEAL (Set Expander for Any Language) ([109] and [110]) is a system that exploited both structured elements in web pages and lexical-syntactic patterns for entity extraction. In the project, the authors addressed the problem of named entity set expansion which is similar to Google Sets. In [109], authors exploited the semi-structured characteristics of web documents to define wrappers which are domain and language independent. The idea of this approach is based on the observation that entities belonging to the same class usually

appear in similar formatting structures on the same web page. For example, if a name of film appears between the HTML tags "<tr><td>" and "</td></tr>", the names of other films will also appear in those similar tags in a web page. These HTML tag patterns were studied from initial entity seed set to extract other entities. To eliminate noise entities, a ranking measure based on graph walk was used to find the similarity between the seeds and extracted entities. This system continued to be improved in [110] in which character-based method similar to Brin ([17]) was combined to improve performance of SEAL system.

Recently, Carlson and colleagues at Carnegie Mellon University have developed a system called NELL ([22]), a never-ending language learner that has ability to extract or read semantic information on the web. This system in the ReadTheWeb project applied a diverse set of extraction methods. Firstly, it deployed a semi-supervised learning method combined with constraints for extracting entities and facts from the Word Wide Web. NELL system also learned contextual patterns from seed examples for extracting instances and relations from sentences on the web. Besides, it implemented a module similar to SEAL ([109] and [110]) to mine semi-structured web data, such as tables and lists on web pages, to extract novel instances of the corresponding relations. The generated instances and relations are then served as the input of a first-order relational learning algorithm to learns probabilistic Horn clauses which are used to infer new relation instances from other relation instances in the knowledge base. According to the report of Carlson et al ([22]), NELL populated a knowledge base with over 242,000 facts in 67 days with an estimated precision of 74%.

2.3.3 Unsupervised approach

In general, unsupervised approach are domain-independent and target different kinds of resources on the Web, including texts, HTML tables, and lists. One method for unsupervised approach to extract relations of entities on web without using human knowledge is to employ the web environment as a corpus to cluster the pairs of co-occurring named entities according to the similarity of the context words between them. This idea was mentioned in the study of Hindle et al in 1990 ([55]). The authors proposed unsupervised clustering techniques to

cluster the noun phrases occurring as subject or object of a given verbal phrase. For example, the word *wine* may occur with *drunk* or *produced*, but not with *eaten* or *driven*. Study of Shinyama et al ([97]) was another work which clustered all possible relations from text and represented them in tables. In these clustering based approaches, relation labels will be manually assigned for each cluster of pairs of named entities. Therefore, it is difficult to apply these techniques to large-scale extraction systems. In order to overcome this limitation, Bollegala et al ([15]) have recently proposed a clustering algorithm to extract relations between entities from unlabelled data. This work clusters simultaneously the entity pairs and vocabularies in different sentences. Due to this process, it can identify representative lexical patterns of semantic relations and use them to propose the relation names for entity pairs. However, relations extracted by the system are not in well-defined representation because they are generated from word phrases between entities in documents.

Beside the clustering based techniques mentioned above, some recent research studies proposed novel methods to extract tables or records from the web environment. According to the study of Cafarella et al in 2008 ([20]), there are over 100 million tables on the web. The meaning of these tables, however, is rarely explicit from their data. Therefore, Cafarella et al ([21], [20] and [19]) developed a WebTables system which exploited tables on the web as a source of high quality relational data for search engines. The statistical information about the co-occurrences of attributes in tables on the web could be used to implement a column thesaurus and propose a column auto-completion in queries. Inspired by the benefits of web tables and in order to furnish more semantics for tables on the web, some recent study have extracted information from web tables. For example, Limaye et al. ([71]) proposed a graphical model for annotating table columns on the web with the labels of types, detecting binary relations, and assigning table cells with entity identifiers from an ontology, such as YAGO ([101]). Meanwhile, Venetis et al ([108]) have recently developed a statistical reasoning model to determine a label for each column and binary relationship in tables.

Moreover, Cortez and colleagues ([31], [30]) have recently proposed a matching strategy to associate text segments in input strings with attributes of a given domain. Firstly, texts are decomposed into segments by using punctuation marks and some heuristics. Next, the

fitness functions are defined to compute the similarity measure between each segment with values of various attributes in the knowledge base of the system. Then, the text segments are assigned to the attributes with the highest score of similarity measures. According to the experiments in the study, this method outperforms CRF-based methods in extraction of attribute values. Furthermore, this method can deal with the cases in which the attributes values in the input string are not in a any particular order. However, the performance of this method significantly depends on the knowledge base of the domain that it currently works on. The fitness functions between a text segment and an attribute in knowledge base are computed by the probability of occurrences of the tokens of the segment in the values of that attributes. As a result, if a token does not appear as a value in a field, the fitness function of that token with that field will be zero. This obviously leads to an disadvantage of this method. The values of a field in knowledge base must cover all tokens that may occur in matching process so that the matching scores between a token with fields can be achieved. This requirement is difficult to obtain in practice when we extract information of entities in large scale application.

According to the study of Gibson et al ([44]), 40–50% of the content on the web is template content. Therefore, the similar structures of web contents on HTML web pages in a site can be exploited to extract information on those web pages. Similarly, this idea can be found in the work of Gulhane and colleagues ([50]) in which the authors proposed a similarity metric that leverages the templatized structure of attribute contents to measure the similarity between two sets of attribute values on web pages. Although several efforts have attempted in the literature to extract information of entities, existing information extraction approaches are not sufficient to provide efficient solutions to the problem because of the variety of noise in the content of web pages.

2.4 Entity Extraction by Text Segmentation

In section 2.2 and 2.3, we have presented the previous approaches for the problem of entity recognition and relation extraction from generally textual documents. In this section, we

focus on the problem of entity extraction on unstructured lists on web documents by text segmentation. In this context, information of entities are organised on implicit records under the form of lists on textual documents.

Information extraction by text segmentation (IETS) is the process of converting an unstructured document which contains implicit records into structured form by splitting the document or a list into substrings which contain data values ([94]). In other words, each text input or document forms one or several implicit records and each implicit record is represented in a form of a list of field values of one or several entities. In general, existing research works on the problem of IETS can be categorised into rule-based approach and learning-based approach.

2.4.1 Rule-based approach

Rule-based approach mainly exploits knowledge of a particular domain to describe data of interest. From a knowledge base in a particular domain, several templates, rules and extractors are generated to extract information of entities. This idea can be found in INFOMAP system ([38]) which employed an ontology of citations, including hierarchical concepts and relations between these concepts in bibliographic domain, to match predefined templates with citations written in six fixed reference styles. Each reference style defines the order of values of entities in the sequences of input texts. However, people may write their citation strings in various styles on their documents and the styles are unknown in advance. Therefore, the ability to adapt to the diversity of reference styles is still a big challenge for this research work.

Flux-CiM system ([31]) is the result of another study that exploits a reference table in citation domain to extract metadata of citations. This method firstly splits each citation string into text segments based on some punctuation marks. Then fitness functions are defined to estimate the probability of a block to be labeled as a field in the reference table. In other words, text segments are mainly labelled by a matching technique. Although this method is robust in extracting citations in different styles, it requires that the vocabulary set of a

field in the reference table must be large enough to cover a representative portion of the domain of interest. Therefore, any bias on the size of vocabularies in a reference table will directly affect the probability of a candidate text segment to a field in the reference table. As a consequence, it influences the quality of the extraction process.

In general, the role of existing knowledge base is essential for rule-based approach. Therefore, building such a huge knowledge base as well as rules to extract information of different types of entities in different domains is one of main drawbacks of this approach.

2.4.2 Statistical learning approach

Meanwhile, previous studies that follow statistical learning approach formulated entity extraction by text segmentation as the problem of labelling a sequence of words. In other words, each word in a sequence needs to be assigned a label to represent a field that the word belongs to. One solution to assign labels for tokens is to view the problem of labelling tokens as the problem of classification in which an extraction model must determine whether a token is assigned a particular label or not. The study of Han et al ([53]) is such research work that used Support Vector Machine (SVM) as a classifier to extract the meta-data of entities in bibliographic domain. In this work, each token is independently assigned a label based on its set of features. Nevertheless, the labels of adjacent tokens are seldom independent of each other. The labels and features of neighbour tokens could be used to determine the label of a token. Therefore, other statistical extraction models were proposed to capture the dependency between the labels and features of adjacent words, such as Hidden Markov Models (HMMs) ([95], [16], [4]) and Conditional Random Fields (CRFs) ([68]). As proven in the study of Peng et al ([85]) and Zhao et al ([119]), CRFs-based methods outperform HMM-based methods in their experimental evaluations for the problem of sequence labelling.

Peng et al ([85]) and Councill et al ([33]) successfully applied CRFs to extract information from research papers and the results of their works were respectively incorporated into two paper search engines REXA[1] and CiteSeerX[2]. However, obtaining a large amount

[1] http://rexa.info

[2] http://citeseerx.ist.psu.edu

of training data, which includes the associations between string segments with their corresponding attributes, to build an extraction model requires a lot of laborious work and may be very expensive or even unfeasible in some situations. Therefore, one of the most challenging tasks for these learning methods is to build such a manual training dataset which covers all possible styles on the web.

Later, some studies proposed the usage of pre-existing datasets to alleviate the need for manually labeled training data ([4], [75], [119]). In those methods, known values in a database are used to train a statistical model for recognising the values of their attributes in an input text. As proposed in the study of Mansuri et al in 2006 ([75]), the authors tried to reduce the dependence on manually labeled training data by exploiting an existing structured database to define some new features. Their method used only few labeled training instances to train a CRFs-based extraction model. Although this method exploited additional features from a structured database, it still needs some user-provided training data in their method.

Meanwhile, Agichtein et al in [4] exploited reference tables to generate training data automatically for training an HMM. This idea was then exploited and improved by Zhao and colleagues ([119]) when they proposed an unsupervised method with CRFs. The order of these field values is firstly inferred by CRFs-based models for attributes. Then, a training dataset is directly generated from a reference table by concatenating attribute instances in the table according to that total order. After that, that training dataset is used to train a global CRFs-based extraction model for extracting information of citations in the input text. However, as the training dataset is directly built from a database, those methods made an assumption about the overlapping of format-related features between field values in knowledge base and input texts. Therefore, the CRFs-based model may not learn enough statistical information of features to label field values in input lists and obtain high performance if the reference tables and the input texts come from different sources ([30]). Moreover, those methods have low performance in running time because they execute inference step and training step for each time they perform a new extraction ([119], [30]).

Recently, Cortez et al ([30], [32]) have proposed ONDUX, an on-demand unsupervised method, to overcome those drawbacks. The authors exploited the high overlapping between

a knowledge base and an input list to segment texts and label them. Then the labels of text segments are used to build an HMM-based graphical model to revise the results. In their work, they implicitly assumed that the majority of labels which are correctly assigned in matching step can help to build a graphical model to rectify incorrect and mismatched ones. Therefore, the method requires high overlap between a knowledge base and input lists so that the statistical model built from matching labels can be generated correctly.

This thesis can be considered as an improvement of ONDUX when we propose several techniques to overcome the technical limitations of previous studies and improve the performance of entity extraction from lists. Firstly, to improve the labelling process, we propose a format-enhanced labelling technique to label data values of entities on input list. Secondly, we exploit the approximately positional information of labels in different sequences in an input lists in a proximity-base positional model to improve performance of graphical model when revising labels. Eventually, the structural similarity between text segments in different sequences in input list is exploited and combined with data-shifting alignment technique to reduce the dependency of knowledge to extraction results. Technical details will be presented in the next chapters of thesis.

2.5 Approximate String Matching

Information about entities is stored on different sources and their data values are often represented as character strings. To detect the duplicate entities on web pages or integrate information of entities from the different sources, matching measures on strings can be necessary for matching entities. However, their information could be written in different formats because of inconsistent representations or input errors. Therefore, approximate string matching algorithms for entity names are indispensable. They also play important role in several problems in database area, for example detection of duplicate records, text searching, entity duplication, etc.

As the typographical variations of entity are common mismatches, we focus on string matching algorithms to deal with typographical issues in this literature review. Other issues

related to phonetic and numeric similarity metrics can be seen in a comprehensive survey on approximate string matching techniques in the study of Elmagarmid et al ([40]). In general, existing similarity measures for string matching take two strings as their input and return a similarity score which quantifies the match between them. The measures for approximate string matching in previous studies can be categorised into two kinds, comprising of character-based and token-based measures.

2.5.1 Character-based measures

In character-based methods, the distance between two strings is defined as the minimal cost to convert a string to the other one by using edit operations such as copying, insertion, substitution, and deletion. Therefore, the measures following this approach are also known as edit-distance measures. Previous studies defined different weights for those edit operations.

Levenshtein's measure in [69] was such a simple one in which the cost of insertion, substitution, and deletion was defined to be one, whilst that of copying was zero. For example, the Levenshtein's edit-distance measure between two strings "Dat Tan Huynh" and "Dat T Huynh" is two because two operations needs to be performed to delete two characters "a" and "n" in the first string to achieve the second string. Let $D(s,t,i,j)$ be the edit distance between first i characters in s and first j characters in t, s_i and t_j accordingly be the i^{th} and j^{th} character of the string s and t_j. The edit distance $D(s,t,i,j)$ is formally defined as in equation 2.6. To implement the Levenshtein's measure, a recursion or dynamic programming technique ([29]) can be applied.

$$D(s,t,i,j) = min \begin{cases} D(s,t,i-1,j-1) & \text{if } s_i = t_j \\ D(s,t,i-1,j-1)+1 & \text{if } s_i \text{ is replaced by } t_j \\ D(s,t,i,j-1)+1 & \text{if } t_j \text{ is inserted} \\ D(s,t,i-1,j)+1 & \text{if } s_i \text{ is deleted} \end{cases} \tag{2.6}$$

That measure was then extended in Needleman and Wunch's measure ([82]) where different costs were assigned for insertion, substitution, and deletion. Similarly, Smith and

Waterman in [98] also utilised edit distance for string matching, but they defined different costs for insertion operation in different positions in a string.

Meanwhile, Jaro's measure ([64]) on two strings was based on the number and position of the common characters between two strings. A character in one string was considered to be in common with a character in the other string if they were the same and their relative positions in the two strings were close enough. Specifically, a character s_i at the position i was in the common string between string s and t if there exists a character t_j at position j such that $i - \frac{min(|s|,|t|)}{2} \leq j \leq i + \frac{min(|s|,|t|)}{2}$. The common string between the two strings is then consisted of those common characters.

Let $s' = s'_1...s'_k$ and $t' = t'_1...t'_l$ be the common strings between s and t and between t and s respectively. The number of transpositions between s' and t' is defined as the number of positions i such that $s'_i \neq t'_i$. The string matching degree was defined on the number of corresponding positions where the characters in the two strings were different, the lengths of the strings, and their common string. Formally, the similarity measure between two strings s and t is defined in equation 2.7.

$$Jaro(s,t) = \frac{1}{3}.(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - \frac{T_{s',t'}}{2}}{|s'|}) \tag{2.7}$$

where $T_{s',t'}$ is the number of transpositions between s' and t'.

In [113], Winkler modified Jaro's measure by taking into account the common prefix between matched strings. Let p be the number of characters in the longest prefix string of s and t, and $p' = max(p, 4)$. The equation 2.8 illustrates how Winkler's similarity between two string s and t is evaluated. As discussed in [27], Jaro's and Jaro-Winkler's measure are good for matching short names.

$$Jaro - Winkler(s,t) = Jaro(s,t) + \frac{p'}{10}.(1 - Jaro(s,t)) \tag{2.8}$$

where $T_{s',t'}$ is the number of transpositions between s' and t'.

Instead of using the common characters in matched strings, the idea of edit distance measures between two strings can be adapted for q-grams, which are short character substrings

of the length q of the matched strings. The similarity between two matched strings will be high value if they share a large number of q-grams in common ([106], [105]). Similar to the ideas of Smith and Waterman ([98]), positional q-gram measure ([103]) was proposed as an extension of q-grams measure when it considered the positions of q-grams in strings. In [46] and [47], Gravano et al showed that positional q-grams measure is effective for approximate string processing in a relational database.

2.5.2 Token-based measures

The idea for token-based measures is that the order of tokens in string matching is not important. For example, "Huynh Tan Dat" and "Dat Tan Huynh" are considered to be the same. This means that the measures in this approach were based on the number of common tokens between two strings. One simple method of this approach was Jaccard's measure, where the similarity between two strings was defined to be the ratio between the number of common tokens and the total number of tokens in the strings (equation 2.9).

$$sim(s,t) = \frac{|S \cap T|}{|S \cup T|} \tag{2.9}$$

Moreover, as proposed in the study of Cohen et al in 1998 ([28]), a string could be considered as a document consisting of terms or tokens. From the point of view of information retrieval, the problem of matching two strings in this case can be considered as the problem of matching two documents. According to vector space model in information retrieval, a document is represented by using a vector over a set of indexed terms in the discourse. A term in the model can be a word or a phrase and it is assigned a weight in the vector. Therefore, when words are chosen as terms in the model, the size of dimensions of the term vector is the number of distinct words in a given collection of documents.

Let N be the total number of documents in the system, n_i be the number of documents where the index term k_i occurs, and $freq_{id}$ be the frequency of k_i, which is calculated by the number of times the token k_i occurring in a document d. The normalised term frequency (TF) of k_i in d and the inverse document frequency (IDF) for k_i is respectively defined by

equation 2.10 and 2.11.

$$tf_{id} = log(freq_{id} + 1) \tag{2.10}$$

$$idf_i = log(N/n_i) \tag{2.11}$$

Based on the properties of the definitions, tf_{id} quantifies the occurrence degree of the term k_i in a document d whilst idf_i measures the significance of the occurrence of k_i in a document. The more the number of documents where k_i occurs is, the less significant the occurrence of k_i is. From the definitions of TF and IDF of a term k_i, the indexed term weight of k_i to a document d is derived and formulated in equation 2.12.

$$w_{id} = tf_{id} \times idf_i \tag{2.12}$$

Let $(w_{1d}, w_{2d}, ..., w_{td})$ be the vector representing a document d and $(w_{1q}, w_{2q}, ..., w_{tq})$ be the vector representing a document q. The similarity TF-IDF of d and q is defined by the cosine of the angle between these two vectors. Formally, it is defined as in equation 2.13.

$$sim(d, q) = \frac{\sum w_{id} \times w_{iq}}{\sqrt{\sum w_{id}^2 \times \sum w_{iq}^2}} \tag{2.13}$$

According to the evaluation of Bilenko et al in 2003 ([14]), TF-IDF is a good token-based measure for string matching. However, this similarity measure does not capture word spelling errors in strings. Therefore, Gravano et al ([48]) extended TF-IDF measure to handle spelling errors by using q-grams instead of words or tokens.

Meanwhile, Bilenko and colleagues ([14]) proposed a Soft-TFIDF measure which modified the TF-IDF method as follows. Let S be a query string and T be a fact string, and $CLOSE(\theta, S, T)$ is a set of tokens u in S which there exists a token v in T such that the similarity between u and v is greater than a threshold θ. Formally, $CLOSE(\theta, S, T)$ is formulated as in equation 2.14.

$$CLOSE(\theta, S, T) = \{u \in S | \exists v \in T : sim'(u, v) > \theta\} \tag{2.14}$$

where $sim'(u, v)$ is a secondary similarity measure between u and v. This secondary similarity measure should be a measure that works well for short strings, such as edit distance ([69]), Jaro's ([64]), or Winkler's measure ([113]).

For each token $u \in CLOSE(\theta, S, T)$, let $N(u, T)$ be the maximum secondary-similarity between u and all tokens v in T, formally $N(u, T) = max_{v \in T} sim'(u, v)$. Then, the Soft-TFIDF matching similarity measure between S and T is defined as in equation 2.15.

$$sim(S, T) = \frac{\sum_{k_i \in CLOSE(\theta,S,T)} w_{iS} \times w_{iT} \times N(k_i, T)}{\sqrt{\sum w_{iS}^2 \times \sum w_{iT}^2}} \tag{2.15}$$

Bilenko et al ([14]) employed Winkler's measure ([113]) for the secondary similarity sim' and chose $\theta = 0.9$ in their experiments. Their experimental results showed that SoftTF-IDF is better than Winkler and TF-IDF in name matching tasks.

2.6 Summary

The main motivation of entity extraction is to capture information about entities on pervasive web data to store them in structured forms. In this chapter, we have provided the background and an overall picture of related problems and solutions in the literature about entity extraction from web data. We have presented entity recognition and relation extraction, which are two typical and important problems in entity extraction. For the problem of entity recognition, rule-based approaches defined a set of rules and focused on the optimisation of rule-matching techniques to perform rules efficiently. A drawback of this approach is that the rules are defined manually and not general enough to extract all entities on web documents. Meanwhile, statistical learning approach for entity recognition requires a manually training data to build an entity extraction model. Since the extraction results are affected by the quantity and quality of manually training data, the application of the approach is limited in web-scale applications.

To extract relationships between entities on the web, previous studies in supervised approach mainly exploits NLP techniques to transform sentences which contain entities into parsed trees. Then the rich features from parsed trees are used to to build training data for

learning an extraction model. Since this approach requires part-of-speech tagging and sentence parsing, they cannot extract information of entities organised in other forms such as lists or tables. Meanwhile, semi-supervised approach mainly exploits generated patterns of given relations to extract information. This approach cannot be applied when we do not know in advance the structure of the patterns for arbitrary relations on the web. Moreover, how to evaluate and control the quantity and quality of the generated patterns in the process of entity extraction to keep obtaining good performance is still a challenge of this approach. Eventually, unsupervised approach for relation extraction exploits clustering techniques to group entity pairs into clusters before their relations can be labelled and extracted. This approach requires that all entities are recognised before their relations can be extracted. That assumption could be difficult to be obtained in practice when relations of entities are extracted from arbitrary sources on the web.

Whilst majority of previous approaches extract information about entities from generally textual documents, this thesis focuses on entity extraction from textual lists, a pervasive and ubiquitous resource which contains information about entities under implicit records in textual documents. We have surveyed related studies on entity extraction by text segmentation and also discussed some string matching techniques for entity names. Moreover, we have reviewed and discussed existing techniques proposed in recent research studies on the problem and we argue that there was no complete system with scalability on entity extraction from textual lists. We also pointed out the novelty and contributions of our approach when compared to existing studies. Eventually, we have reported the majority of the survey in this chapter in our publication in [58]. In the next chapters, we will present our proposed framework to tackle the problem of entity extraction by text segmentation and our technical contributions when we solve the problem.

Chapter 3

Self-Supervised Learning Framework

3.1 Introduction

As surveyed in section 2.4, designing an application that can automatically recognise and parse information of entities from textual lists scattered over the internet is not a trivial task. It is also an important and practical research problem in the literature. Since field values in input lists are expressed in a textual representation, traditional wrapper-based methods ([35], [7]) cannot be applied to the inputs that are formatted differently in HTML. Instead, the problem of entity extraction from textual lists is formulated as a problem of text segmentation to extract data values in them. Each sequence in a list is split into text segments which are field values of entities. In general, a dominant approach for this problem in the literature is the deployment of statistical methods to assign labels for text segments and extract field values of entities from sequences in input lists. Previous studies deployed two different techniques to generate training data for building statistical extraction models. The first technique, which is called supervised approach, builds training data manually ([95], [43], [16], [68], [85], [75]). Those supervised methods require a large amount of user-provided training data, which consists of a set of text segments, their features and labels. Obtaining such a training dataset to extract several types of entities on different domains on the web written in any styles with different orders of field values and punctuation marks may cost a lot of time

and laborious work. In addition, the generated extraction models are difficult to be reused to deal with different types of inputs on the web.

Meanwhile, the second technique, which is called unsupervised approach, exploits the pre-existing data source to alleviate the need for labelled training data when building extraction model ([4], [119], [30]). In the study of Agichtein et al ([4]), the training datasets are built automatically by concatenating field values and their labels in a reference table before it is used to train an Hidden Markov Model (HMM) ([16]) for information extraction. Later Zhao et al in [119] adapted this idea to Conditional Random Fields (CRFs) ([68]). Since training datasets are generated directly from a reference table by concatenating its field values, the methods made a strong assumption about overlapping features between field values in a referenct table and input lists. According to the experiments of Cortez et al in [30], the quality of extraction in the study of Zhao et al ([119]) is quite low when input lists and referent table come from different sources. Moreover, the running time of both methods [4] and [119] is slow because they need to infer the order of field values, generate training data and construct a new extraction model when they perform a new extraction. Our proposed method is similar to their methods when we also exploit knowledge base to automate our extraction model. However, instead of building training data directly from field values in a knowledge base, we exploit the knowledge base to capture the structure of input lists in a statistical model to extract information.

A recent study which is close to our work is ONDUX, a study of Cortez et al in 2010 ([30]). In their work, overlapping terms between knowledge base and input lists is exploited to label text segments and build a statistical model for information extraction. Their method mainly exploits high overlapping terms between a knowledge base and an input list. The study has not considered a format similarity between field values in a knowledge base and and an input list, the distribution of positions of field values as well as their structural similarity within a list to enhance the quality of information extraction. Our work can be considered as an improvement of ONDUX whilst we identify those issues, address research problems and proposed technical solutions to improve performance and overcome the limitations of the system.

In this chapter, we firstly present our proposed framework that automatically constructs training data from web lists and build a graphical model for a parser to extract information about entities. The basic idea in our framework is that we exploit an existing knowledge base of entities to label text segments on sequences of textual lists and use statistical information of the labels to capture the implicit structure of the input lists and train a statistical extraction model for parsing information about entities in the input lists. There are a lot of types of knowledge base that can be employed in the framework. They can be tables, relations, or ontologies, etc. Moreover, we propose a format-enhanced labelling technique which exploits the dyad of membership relation between a field value and its label in this chapter. Due to this model, we can incorporate both format and content-based matching techniques to improve the quality of extraction model.

The structure of this chapter is organised as follows. In section 3.2, we formally define the problem of entity extraction by text segmentation. Then, section 3.3 describes our proposed self-supervising learning framework which exploits the overlap between a knowledge base and input lists to construct automatically a statistical extraction model for extracting data values of entities from the lists. Next, our proposed format-enhanced labelling technique is presented in section 3.4. After that, experiments and evaluations are illustrated in section 3.5. Eventually, section 3.6 summarises some remarks of the chapter.

3.2 Terminology and problem definition

In this section, we present a definition of the problem of entity extraction from lists and the terminologies we utilise in our study. Firstly, we consider a list L of n sequences. Each i^{th} sequence l_i in the list L is a string which represents an implicit structured record. For example, in bibliographic domain, each sequence l_i is a reference or a citation, which is a string to represent the field values of an academic publication. A typical representation of a publication may include information about authors, title, book title or publication venue, pages, date, volume, and some other information of a publication.

Formally, each a sequence l_i can be represented as $l_i = \{v_1 d_1 v_2 d_2 v_3 d_3 ...\}$, where v_j

are *field values* of an implicit record in the input list L, and d_i are *delimiter* or symbols to separate field values v_i. A *delimiter* is any character other than A..Z, a..z, or 0..9. The delimiters split a sequence l_i into a set of *text segments* and each text segment can be a *token* or a sequence of *tokens*. We note that the delimiters separate field values but they may occur in field values. Therefore, each text segment can be a field value or several text segments compose a field value. Moreover, the number of field values in different sequences in a list can be different. For example, some citation strings include complete information such as author, title, publication venue, date, location, and publisher whilst other ones use only a subset of those fields.

To assign labels for text segments in input lists, we exploit a knowledge base which contains a set of pairs $K = \{<t_1, E_1>, ..., <t_m, E_m>\}$ where each t_i is a distinct field which is used to assign labels for text segments, E_i is a set of field values or occurrences of the field t_i. Given a list L of n sequences including field values on a textual document and a knowledge base K, the goal of the problem of entity extraction by text segmentation is to extract automatically the field values in L and store them in a structured form, e.g a table, or an xml file.

3.3 Self-supervised learning framework

3.3.1 Framework overview

In this section, we present our proposed self-supervised learning framework for entity extraction task from web lists. Similar to other unsupervised methods, we are interested in automating the process of extracting information about entities from textual lists. As introduced in section 3.1, the intuitive idea in our methodology is to employ a knowledge base to capture the structure of an input list and incorporate a statistical learning method to automatically extract field values in input lists. Whilst supervised approach requires human to build training data or supervise the training extraction model, our method is similar to other unsupervised approach when it builds its extraction model automatically. However, our proposed

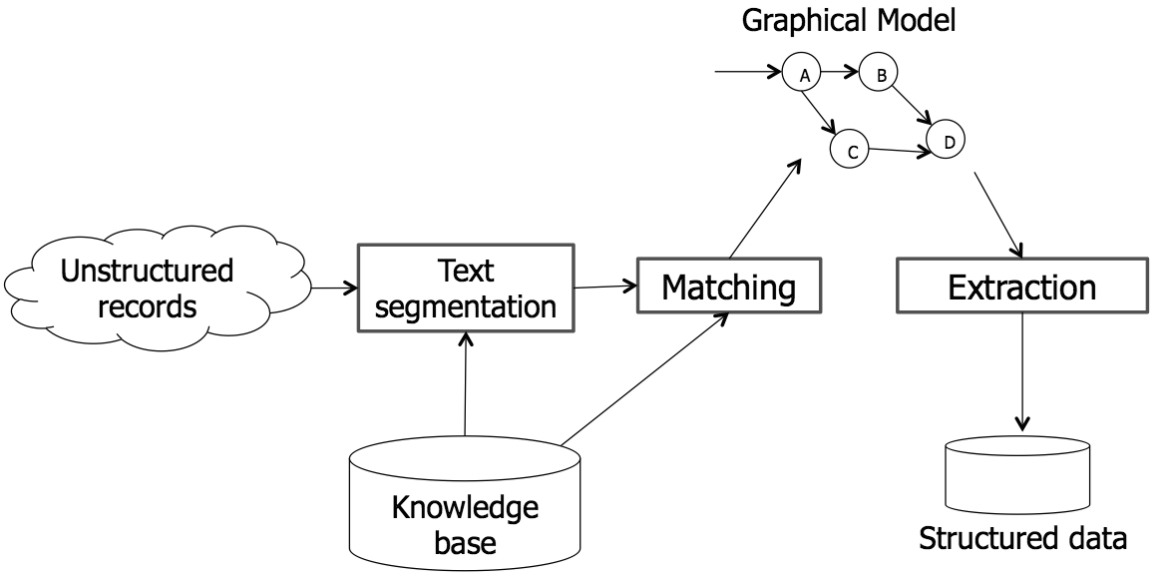


FIGURE 3.1: Self-supervised learning framework for entity extraction from lists

framework is called self-supervised because it can repair and fix failures via a refinement step in the framework when it extracts information about entities.

Figure 3.1 describes the overall architecture of our proposed framework. In general, the framework can be described in a sequence of operations which are basically grouped into three main phases: text-segmentation phase, labelling or matching phase, and extraction (or refinement) phase. Detailed descriptions of those phases are presented in sections 3.3.2, 3.3.3, and 3.3.4, respectively.

3.3.2 Text-segmentation phase

In text-segmentation phase, each sequence in an input list is split into multiple text segments which are then labelled in the next phase. Each text segment is a term or a sequence of terms (or tokens) that constitute a field value of a certain attribute. The simplest way to perform the segmentation step is to use delimiters or punctuation marks to split sequences. As mentioned in section 3.2, the delimiters that separate field values may occur in field values. Therefore, a field value may include one or few consecutive text segments. To improve the quality of the segmentation phase, a knowledge base is exploited. If any two text segments co-occur in

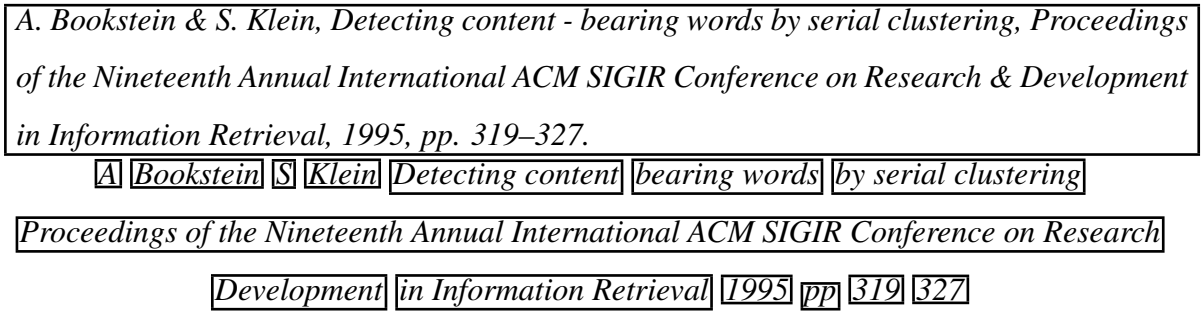


FIGURE 3.2: An example of text-segmentation phase

the same attribute value of a knowledge base, they will be merged into one text segment.

An example of this phase can be seen in Figure 3.2. In the example, the input string is split into several text segments which are words or tokens. Then, the segments are merged together by using a knowledge base. In the example, the text segments in the result are marked in rectangles.

After this phase, we obtain a set of text segments that constitute field values. We notice that text segments may not be a field value at this stage and they will be assigned labels and merged in the next phases. In general, this segmentation step is basically similar to previous studies in supervised learning methods when they manually assign labels for tokens or text chunk in their approach. The only difference in our work is that we exploit a knowledge base to merge text segments to assign labels in next matching phase.

3.3.3 Matching phase

This matching phase is also referred as a labelling phase in our work. In this phase, we assign labels for text segments obtained from the previous phase by exploring a knowledge base. Each label represents an attribute of an entity or a concept in the knowledge base. Given a text segment, the purpose of this phase is to assign a label for the text segment by choosing a candidate label from a set of labels in a knowledge base. Therefore, we need a similarity measure to evaluate how likely a label should be assigned to a text segment and then the measures between labels and a text segment are used to rank and choose the label with the highest score.

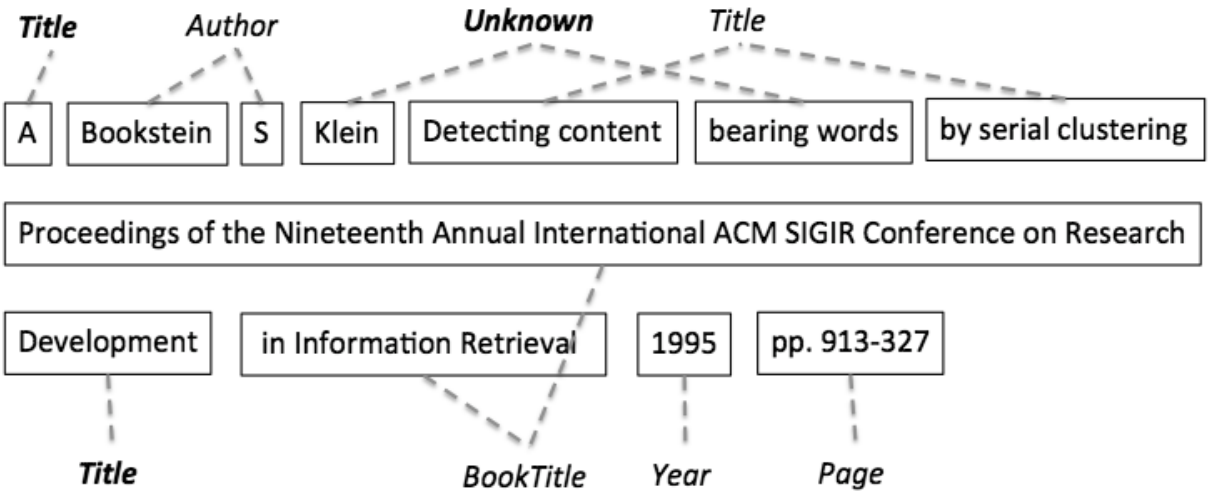


FIGURE 3.3: An example of matching phase

Cortez et al ([30]) adapted the idea of Mesquita et al ([78]) into their study and defined a matching score between a field value and a label t_j via a fitness function as in equation 3.1. The fitness scores are computed for all tokens w in the text segment s and the label t_j and each fitness score is computed as in the equation 3.2.

$$M(s, t_j) = \frac{\sum_{w \in s} fitness(w, t_j)}{|s|} \tag{3.1}$$

$$fitness(w, t_j) = \frac{freq(w, t_j)}{freq(w)} \times \frac{freq(w, t_j)}{freq_{max}(t_j)} \tag{3.2}$$

where $freq(w, t_j)$ is the number of values of the label t_j containing the token *w*, $freq(w)$ is the total number of instance values in the knowledge base containing the token *w*, and $freq_{max}(t_j)$ is the highest frequency of any token in the instance values of the label t_j.

In our study, we reuse the matching score function defined in the study of Cortez et al ([30]). We also notice that the matching score function in [30] only focuses on the content-based similarity between text segments and field values in a knowledge base. However, not all text segments can be matched in a knowledge base. Instead, format-based similarity between text segments in an input list and field values in the knowledge base could be exploited to improve the performance of matching phase. We will present our improvement in this phase in the section 3.4 of this chapter.

An example to illustrate the results of this matching phase can be seen in Figure 3.3. The example demonstrates the results of matching phase of the previous example in Figure 3.2. The labels obtained in this labelling phase are then utilised to build a statistical model for extracting information about entities from web lists in the last extraction phase of the framework. We also notice that the labels obtained from this phase are not always accurate. Some labels may not be matched in knowledge base and they are not assigned to any labels. Similarly, some other labels could be mismatched. For example, the labels for the text segments *Klein* and *bearing words* are *Unknown* whilst the text segment *Development* has the label *Title* which is not correct. Those such labels will be revised in the extraction phase, which is the next phase of the framework.

3.3.4 Extraction phase

The main purpose of this extraction phase is to exploit statistical information of the labels obtained in the matching phase to build a graphical model to revise the results of the matching phase and extract information about entities. The key idea of this phase is that the labels of texts segments in an input list reflect statistical information of the distribution of field values in an input list. Such information could be exploited to assign the labels for unmatched segments and rectify mismatched ones. This strategy is based on an assumption that the number of correct labels are more than incorrect ones in an input list. Therefore, statistical analysis on labels enables us to detect incorrect ones and fix them.

The purpose of the graphical model in our framework is to capture the structure of sequences and represent the likelihood of transitions between labels as well as the occurrence of a label in an input list. We define a positional and sequential model (PSM), which includes a sequential model (SM) and a positional model (PM), to revise the labels in refinement phase. Whilst the sequential model is to capture the likelihood of transitions between labels in an input list, the positional model to reflect the possibility of the occurrence of a label in a particular position in the list. For example, if there are several transitions from the label "author" to the label "year", the probability to revise an unknown label before the label "year"

should be higher than probability to have other labels. Similarly, if a label occurs frequently in a particular position, it will high possibility that an unknown label at that position should be revised to assign the label.

A PSM for labels is defined by the following three components:

- A set of states $T = \{start, t_1, t_2, ..., t_N, end\}$ where $start$ and end are respectively initial state and ending states and each state t_i represents a label of a text segment.

- A matrix A where each element a_{ij} is the probability of making a transition from state i to state *j*. Each element a_{ij} in the matrix A is defined as the equation 3.3. The matrix A is referred as a sequential model for the transitions between labels in an input list.

- A matrix P where each entry p_{ik} denotes the probability of the label t_i appearing in the position k^{th} in an input list. Formally, p_{ik} is defined as in the equation 3.4. Similarly, the matrix P is referred as a positional model for labels in an input list.

$$a_{ij} = \frac{Number\ of\ transitions\ from\ state\ t_i\ to\ state\ t_j}{Total\ number\ of\ transitions\ out\ of\ state\ t_i} \tag{3.3}$$

$$p_{ik} = \frac{Number\ of\ observations\ of\ t_i\ in\ k}{Total\ number\ of\ segments\ in\ k} \tag{3.4}$$

The transition model can be visualised as a directed graph where each node is a label or state in the model and each edge describes the transition between labels. Each edge in the graph is weighted by the possibility of making a transition from a state to the other state. An example of a sequential model in bibliographic domain can be seen in Figure 3.4.

Since the sequences of an input list may contain different number of text segments and labels, the usage of sequential model (SM) helps to improve the recall of extraction results but it could decrease the precision of the system. Therefore, a positional model (PM) is combined into a PSM model to revise labels in the results of matching step. The idea of using a sequential model and a positional model to capture the structure of input data was originally exploited in a machine learning technique in [16]. However, the main difference

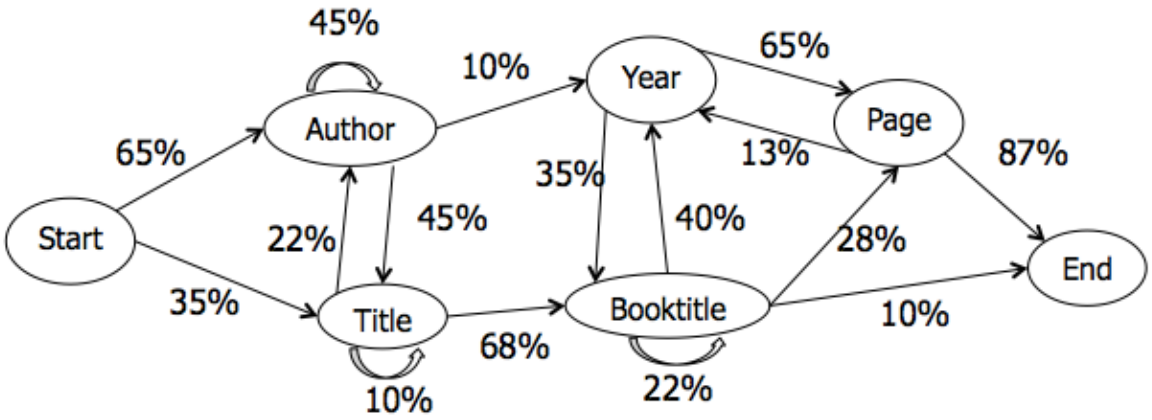


FIGURE 3.4: An example of a sequential model (SM)

between our work and the previous study is that the graphical model in our framework is built from labels in input lists which are dynamically assigned in labelling phase rather than pre-defined manual training data. Moreover, we also note that the position model currently defined in the equation 3.4 only exploits the occurrences of a label in a particular position in a list. We will present our extension for the positional model by exploiting proximity-based information of labels in next chapter of this thesis.

To compute the probability to have a label t for a text segment, matching score, sequential score, and positional model score can be combined by using a Bayesian disjunctive operator, also known as Nosy-OR-Gate ([84], [30]), as in equation 3.5.

$$sim(s,t) = 1 - (1 - M(s,t)) \times (1 - a_{ij}) \times (1 - p_{ik}) \tag{3.5}$$

where $M(s,t)$ is a matching score between a segment s and a label t; i is the index of the label t in a list of labels T, j is the index of the label of the next segment of s; and k is the position of s in an input sequence. The value of a_{ij} and p_{jk} are respectively defined by the sequential model as in equation 3.3 and the positional model in equation 3.4. Both matrices A and P are built directly by a single pass on an input list.

3.4 Format-enhanced labelling technique

As presented in section 3.3, the proposed framework for entity extraction from lists mainly exploits the labels which are assigned at matching phase to build a graphical model for information extraction. The model captures statistical information of labels and structure of input list to revise the labels in matching phase. As a consequence, any mismatched labels or unknown labels may affect the statistical information embedded in the graphical model and the quality of information extraction phase. Therefore, the accuracy of matching phase to assign labels for text segments may play crucial role in the framework.

Given a text segment s, the purpose of the labelling phase is to assign to s a label t from a knowledge base K based on the features of the string s. Each label t can be any dummy text which represents a field or an attribute of entities in the knowledge base K. This labelling task is complicated because exact string matching criteria on data in knowledge base might lead to only few matches or no match, whilst soft criteria might generate a few wrongly labelled strings which will severely impact the accuracy of the statistical extraction model built on those labelled data.

Formally, given a set of records R with the attributes or labels t in a knowledge base K and a set of sequences L, we need to find a set of labels/attributes of t that match with the candidate field values in R. It is necessary to define a soft scoring function $SoftScore(s, t)$ which returns a value in [0, 1] and measures the similarity between a candidate text segment s in a sequence S and the field values of the attribute t in the record R. One of way to implement this scoring function is to consider the text segment as a query on the knowledge base K and apply ranking functions in information retrieval as similarity measure to assign labels. The overlapping terms between knowledge base and the text segment can be exploited to measure the similarity between the text segment and the label. This idea can be found in the study of Cortez et al ([30]) when the authors define the score function by counting the number overlapping terms between text segments and knowledge base.

In our study, we identify the dyadic representation of semantic relations between a candidate field value and a label in knowledge base. We formulate the process of assigning a label

t for a candidate field value s as the process of checking whether the element s is a member of a set with the label t or not. According to set theory, a membership relation between an element and a set can be described in two ways: intensional and extensional definition. Intensional definition of a set formulates its meaning by specifying all properties which an element must satisfy to reach the definition. For example, a year often has four digits and starts with the prefix "19" or "20". Similarly, the page numbers of a paper often starts with one of strings in the set {"page", "pages", "p", "pp", "pg"} and two numbers separated by a punctuation such as "-" or "~". The properties to define a set in the intensional definition can be encoded in rules to verify whether an element is a member of the set or not.

We automatically learn the sequences that represents the format of the label t from a knowledge base and then apply to check the format of a candidate value s. We firstly split each value of t into tokens by using white-spaces. Then we adapt the idea in the study of Borkar et al ([16]) to encode the tokens into a sequence of *symbol masks*. However, instead of building a training data manually, we exploit the existing values in the knowledge base to build masks automatically. The representation of *symbol masks* is to capture the format of values in a knowledge base. For example, the date "25 Aug 2012" is encoded "[dd] [A-Z][a-z][a-z] [dddd]", where the mask [d] represents a digit in from zero to nine, the mask [A-Z] represents an uppercase letter, and the mask [a-z] represents a lowercase letter. From that representation, we generate a transition model m_t for those masks. In the model, each node represents a mask and an edge between two nodes represents a transition of two masks in the values of label t. Each edge in the model has a weight which is defined by a likelihood to transit from a mask to another mask. The likelihood is formally defined in equation 3.6.

$$w(m_i, m_j) = \frac{Number\ of\ transitions\ from\ mask\ m_i\ to\ mask\ t_j}{Total\ number\ of\ transitions\ out\ of\ mask\ m_i} \tag{3.6}$$

Due to the representation of the model, a value of the label t can be described by a path in the model. Similarly, a candidate value s is also encoded into a mask by using the symbol taxonomy as above. Then, the format similarity between a candidate value s and a label t is evaluated as in equation 3.7.

$$f_r(s,t) = \frac{\sum_{<m_i,m_j>\in path(s)} w(m_i,m_j)}{|path(s)|} \tag{3.7}$$

where $path(s)$ is a sequence of masks generated for s. Moreover, we note that if there is no matching mask paths of s and t, the value of $f_r(s,t)$ is equal to zero.

We encode those mask paths as regular expressions for each label t to segment and label them. By using masks to represent the data values of a label t, we can formulate the format-related feature of each label into a model and use it to capture the property of some datatypes such as numerics, dates, URLs, emails, and telephone numbers.

Meanwhile, extensional definition describes a set by specifying every element in the set. In this way, we consider each label t as a set and its field values are elements of the set. Therefore, the similarity between a string s and a label t can be computed via the instance values of the label t in a knowledge base. Based on the values of the labels, we exploit the common tokens/q-grams shared between a string s and the values of a label t to define a similarity function between them. As mentioned in 3.3.3, there are several ways to define this function. One of ways is to view each text segment s as a query and all tokens in the instance values of the label t as a document, then a ranking function in information retrieval can be applied to measure and rank a label t by the relatedness between the query s and the instance values of the label t.

To measure the content-related similarity between a candidate value s and a label t, we reuse ideas in the studies of Mesquita et al ([78]) and Cortez et al ([30]) to define a matching function for this content-related feature as in the equation 3.8.

$$f_c(s,t) = \frac{\sum_{w\in s} fitness(w,t)}{|s|} \tag{3.8}$$

The fitness scores are computed for all tokens/q-grams w in the query string s and the label t and each score is computed as in the equation 3.9.

$$fitness(w,t) = \frac{freq(w,t)}{freq(w)} \times \frac{freq(w,t)}{freq_{max}(t)} \tag{3.9}$$

where $freq(w,t)$ is the number of values of the label t containing the token w, $freq(w)$

is the total number of instance values in the knowledge base containing the token w, and $freq_{max}(t)$ is the highest frequency of any token in instance values of the label t. The first fraction in the equation 3.9 represents the probability to have the token w in the type t. Given a token w, the first fraction returns the same value if the frequencies of the token in instance values of two different labels are alike. Therefore, the second fraction is used as a normalisation factor to take into account the importance of a token for a label. A token will be more important for a label if it occurs in several instance values of that label as compared to other tokens. Moreover, to implement the fitness function, we adapt the idea of using inverted lists in the study of Chandel et al ([24]) to index all tokens and field values in a knowledge base. Each entry in the list is a token and it is mapped to a list of field values as well as their labels that contain the token. By that way, we can compute quickly the frequency of a token which the instance values of a label t contain.

Due to the dyadic representation of a member and a set, we can incorporate both format-based similarity and content-based similarity into our similarity model in our matching phase. We also note that our previous study in [59] also exploited our proposed format-enhanced matching technique for labelling text segments on web lists. The labelling process in the study is mainly based on the rules which approximately match a candidate text segment s and the aliases of a field value by a similarity score such as an edit distance ([69]). Our later work in [61] improved this measure by combining format-related feature and content-related feature into a single similarity model. In our study, each label is viewed to represent a set and each text segment is considered as a member of a set in set theory. We exploit the intensional and extensional definition of a set to define a membership relation between a member and its set in a formal way. Moreover, our proposed technique can be considered as an improvement the study of Cortez et al ([30]). Whilst the matching phase in the study of Cortez et al ([30]) focuses on overlapping terms to define matching functions between text segments and labels, we combine both format-related feature and content-related feature in our matching process when we match text segments and labels. Due to this improvement, we can achieve high performance in matching phase and hence we obtain high performance when performing the final refinement phase of the framework.

The process of splitting text and matching phase for each sequence in an input list is described in Algorithm 1. We recognise some primitive datatypes including numbers, date-times, page numbers, volumes and issues, URLs, email addresses, and phone numbers, then define regular expressions which are leant from masks of field values in knowledge base and use them to segment texts. Due to this, we can obtain high performance on those simple datatypes and helps achieve effectiveness when we revise the labels of other field values in the input list in the refinement phase. In the algorithm of splitting a sequence into text segments, we initially extract tokens from a string l based on the occurrence of white spaces (line 1). For each token t' in a string l, we find a sequence of consecutive tokens starting from t' which satisfies any regular expressions to group them into a text segment (lines 5-12). Moreover, if any two consecutive tokens co-occur in the same instance value according to a knowledge base, they will be grouped into the same text segment (lines 13-19). Initially, each text segment has a single token (line 13). Then we find if any instance e in the knowledge base K that contains two consecutive tokens or not (line 14). If there exists such an instance (line 15), we put them into the same group (line 17). The process is repeated until all tokens are considered.

Given an input list l_i of an algorithm, the number of regular expressions for format-related features are fixed when we perform the text-splitting phase. For each token t' in each string l_i of L, we need to find a sequence of consecutive tokens starting from t' by using the regular expressions. In the worst case, the statements to find such a sequence (lines 5-12) performs $O(n)$ time, where n is the number of tokens in the string l_i. Since we need to traverse all tokens on the string l_i, the complexity of Algorithm 1 is $O(n^2)$, where n is the number of tokens in string l_i.

3.5 Experiments

The purpose of the experiments in this chapter is to evaluate the performance of our proposed labelling technique when it is used in our framework. We perform experiments on datasets in different domains and compare with the results obtained from previous studies. In this

Algorithm 1: Algorithm for text-splitting phase

Input: A string l in an input list L, a knowledge base K, a set of regular expressions R for format-related features

Output: A set B of text segments

$T' : <t'_0, ..., t'_n>$ = Extract all tokens from string l
j = 0, i = 0
while $j < n$ **do**
    k = j + 1
    while $k < n$ **do**
        if $t'_j ... t'_k$ *satisfies a regular expression in* R **then**
            $B_i = \{t'_j ... t'_k\}$
            i = i + 1
            j = k +1
        end
        k = k + 1
    end
    $B_i = \{\} \cup <t'_j>$
    $C = \{<t, E> \in K, e \in E | t'_j, t'_{j+1} \in e\}$
    if *C is not empty* **then**
        t'_j and t'_{j+1} co-occur
        $B_i = B_i \cup <t'_{j+1}>$
        j = j + 1
    end
    i = i + 1
    j = j + 1
end

section, we firstly describe the datasets, experimental setups and metrics that we employ in our evaluations. Then we report experimental results and compare them with the existing study.

3.5.1 Data settings

We perform experiments on public datasets in two domains: *bibliographic* and *addresses*. In each domain, we build a knowledge base and testing data from the data sources. In *bibliographic* domain, we use *CORA* collection ([85]) and *PersonalBib* dataset [75, 119] from experiments in previous studies. In *Addresses* domain, we download two datasets *BigBook* and *LARestaurants* from the RISE repository ([3]) and then manually label the field values in each dataset. Detailed information about those datasets in both domains is summarised in table 3.1. In the experiments, we verify the extraction results in each phase and evaluate how much our proposed techniques can give better performance than the results obtained by existing studies on the problem of information extraction by text segmentation.

Domain	Dataset	Number of attributes	Number of records
Bibliographic Data	*Cora*	13	500
	PersonalBib	7	395
Address Data	*BigBook*	5	4,000
	LARestaurants	4	250

TABLE 3.1: Domains and datasets used in our experiments.

3.5.2 Metrics for evaluation

In order to assess the experimental results, we utilise well-known precision, recall, and F1 measures ([74]) in our comparisons. We denote A_i as a referent set and B_i as testing results to be compared with A_i. The precision (P_i), recall (R_i) and F1 measure (F_i) are accordingly defined as in the equation 3.10, 3.11, and 3.12. In our experiments, A_i is a set of tokens

which compose values with a label and B_i is a set of terms assigned to a corresponding label by our method.

$$P_i = \frac{|A_i \cap B_i|}{|B_i|} \tag{3.10}$$

$$R_i = \frac{|A_i \cap B_i|}{|A_i|} \tag{3.11}$$

$$F_i = \frac{2 \times P_i \times R_i}{P_i + R_i} \tag{3.12}$$

3.5.3 Extraction quality and evaluation

Tables 3.2 describes experimental results in *Bibliographic* domain when we conduct experiments on *Cora* dataset with the usage of *PersonalBib* dataset as a knowledge base. In the experiments, we compare our proposed type-based labelling technique with a content-based labelling technique in the study of Cortez et al ([30]). The column *Ondux-Matching (M1)* in the table refers to the results obtained when we apply content-based labelling technique whilst the column *Our Matching (M2)* refers to the results after we employ our proposed type-based labelling technique. The columns "M1+PSM" and "M2+PSM" present the gains in quality achieved when we utilise PSM model to revise the results obtained from the matching steps M1 and M2, respectively.

We note that the results obtained with content-based labelling technique are similar to those reported in the experiments of ONDUX in the study of Cortez et al ([30]). We include the results here for completeness and to validate the baseline of our experiments. However, we obtain high performance of extraction on simple data types such as page numbers, years, volumes and issues as compared to ONDUX. It can be observed from the table that when we incorporate format-related features into matching phase, we can achieve high performance on the quality of information extraction on the data types with above 94% of F1-measure.

In *Addresses* domain, we exploit *LARetaurants* dataset as testing data and *BigBook* dataset as knowledge when we perform experiments. The experimental results are reported in table

Field	Ondux-Matching (M1)	Our Matching (M2)	M1+PSM	M2+PSM
Author	0.7080	0.7080	0.7822	0.7845
Title	0.7882	0.7882	0.8154	0.8217
Booktitle	0.7971	0.7971	0.7922	0.7967
Pages	0.7546	0.9961	0.8528	0.9961
Year	0.7814	0.9912	0.8990	0.9912
Volume	0.8538	0.8483	0.9578	0.9404
Issue	0.8427	0.9663	0.9263	0.9663

TABLE 3.2: Experimental results on *Cora* dataset using data from *PersonalBib* source.

3.3. The experiments show that the quality of extraction on the dataset is high and we can obtain 98.11% of F1-measure by exploiting format-related features for recognising phone numbers.

Field	Ondux-Matching (M1)	Our Matching (M2)	M1+PSM	M2+PSM
Name	0.6182	0.6182	0.9724	0.9724
Street	0.9073	0.9073	0.9808	0.9808
City	0.7388	0.7388	0.9857	0.9857
Phone	0.862	0.9811	0.9882	0.9923

TABLE 3.3: Experimental results on *LARestaurants* dataset using data from *BigBook* source.

Moreover, we also report that [59] is another case study of our framework which focuses on bibliography domain and we build training data from web lists extracted from web pages. In the study, we used the publication titles of DBLP records to search on the web via some keyword-based search engines such as Google or Bing. Then a set of hyperlinks on returned web pages is extracted before they are used to download and extract the lists of references of the publications on the web pages. After that, the overlapping terms between a knowledge base and the lists are exploited to build a statistical extraction model. The experimental

results in the study also confirmed that the processing time of our method is much better than the running time of U-CRF, which dramatically increase when we increase the size of reference table.

3.6 Summary

In the chapter, we have presented our proposed self-supervised learning framework for extracting information of entities from textual lists on the web. The basic idea of our framework is to exploit the overlap between a knowledge base and a textual list to capture the structure of the lists via the transitions and positions of labels in a list. Due this structure, we can build a graphical model for information extraction automatically. We firstly label information of entities on web lists by using a knowledge base, then the distribution of labels in input lists are then exploited to construct a statistical extraction model automatically.

To label web lists from a knowledge base, we have proposed a format-enhance labelling technique to match candidate field values and labels in a knowledge base. We identify and exploit both intensional and extensional definition of a set in set theory to measure how a text segment should be assigned a label in a knowledge base. By this way, we can incorporate both format-based and content-based measure in a similarity model for labelling input lists. In the experiments, we have demonstrated the effectiveness of our labelling technique as compare to ONDUX ([30]), a state-of-the-art method for information extraction by text segmentation. Whilst ONDUX only focuses on the content similarity between knowledge base and input lists, we accommodate format-related features in our similarity matching model to improve quality of extraction. We have conducted experiments on datasets on different domains to verify results and provide detailed explanations for the extraction results we have obtained. The ideas and research results of this chapter are partially presented in our publications in [59] and [61].

Although our proposed framework can generate a training data automatically for constructing a statistical model to extract information about entities from textual lists, it still has

several gaps that can be improved. Currently, the performance of the framework mainly depends on the overlap between the format and the content of knowledge base and input lists. Meanwhile, information about entities on a list is often written in some particular formats which define some orders of values in a list. This means that the structure of input lists could be exploited to improve the performance of information extraction system on lists.

In the next chapter, we will present a proximity-based positional model for labels to capture positional information of labels in input lists. The proposed model exploits the distribution of labels and their positional information and it is combined with the transitions between labels in an input list to improve the quality of information extraction phase. We will investigate and demonstrate the idea of our proposed techniques in chapter 4 of this thesis.

Moreover, the format-enhanced labelling technique proposed in this chapter is also a initial foundation for our idea to exploit a format-based similarity for grouping similar text segments in different sequences of an input list into clusters. This will help to improve the performance of the labelling process and reduce the dependence on the knowledge base. This will be presented in chapter 5 of this thesis.

Chapter 4

Proximity-based Positional Model for Labels

4.1 Introduction

This chapter introduces our proposed proximity-based positional model to capture positional information of labels in an input list to improve the quality of entity extraction by text segmentation. The basic idea of this model is to capture and exploit approximately positional information of field values in an input list to provide an evidence on the exist of an unknown label in a particular position. The major contribution of this chapter is a proximity-based positional model for labels within a list which considers the distribution of a target label in different positions to measure how likely a label occurs at a particular position. Due to the model, we can improve the performance of the refinement phase in our proposed self-supervised learning framework.

As mentioned in chapter 3, the basic idea of our proposed self-supervised learning framework for the problem of entity extraction by text segmentation is to exploit the overlapping between a knowledge base and web lists to build a statistical extraction model. The text segments are firstly labelled by matching them with a knowledge base, then their labels are revised by a graphical model. Meanwhile, information of entities on the same textual list

is often written in similar style which defines the order of field values on the sequences of the lists. Therefore, a graphical model to revise the labels of text segments should capture transition feature as well as positional information of text segments in an input list. Both transition and position related features in a list provide more evidence on the existence of a label an input list when we build a graphical extraction model. For example, if there were a lot of transitions from the labels "author" to the label "year" in a list of reference in bibliographical domain, it would be high possibility that the text segment with unknown label before the label "year" should be assigned as "author" although it was not matched with any field value in the knowledge base. Similarly, if the label "author" occurs frequently in the first position of a reference list, an unknown text segment in the position should be assigned as "author".

Following that idea, Cortez et al ([30]) proposed a combination of HMM-based positioning and sequencing model to build the graphical model. However, they only exploited fixed positions of a labels in different sequences of input list. It means that the positional model in their work is defined by considering the number of occurrences of a label in a particular position of different sequences of an input list.

We argue that although the field values in different sequences of an input list could be written in similar styles, the labels of text segments in an input list which are assigned by using a knowledge base could not always be in a fixed positions in different sequences. It cannot be ensured that majority of correct labels always occur in a fixed position in different sequences of an input list. Therefore, it would be rigid if we only consider a fixed position of a label and do not consider the distribution of the label in an input list. Instead, we should accumulate the distribution of the labels in different positions in the input list.

As an example to illustrate the issue, let us consider consider the labels "volume" in reference strings in bibliographic domain. Even they are written in the same order of field values, the number of authors in those strings can be different and the lengths of paper titles and book titles can also be different. Therefore, the positions of the label "volume" in different sequences of an input list can be different. Another example can be seen in Figures 4.1 and 4.2. The figure illustrates an example of the results when we apply a labelling phase

Positions of segments	1	2	3	4	5	6	7	8	9
Line 1	**Author**	Title	**Author**	Title	Title	Booktitle	BookTitle	Year	Page
Line 2	?	**Author**	?	**Author**	Title	?	BookTitle	?	Page
Line 3	?	Title	**Author**	Title	Title	?	BookTitle	Page	
Line 4	**Author**	**Author**	?	Tittle	BookTitle	?	?	?	Year
Line 5	?	?	Title	Title	BookTitle	BookTitle	BookTitle	Year	?

Author?
Title?

FIGURE 4.1: An example of labels to be revised after matching phase

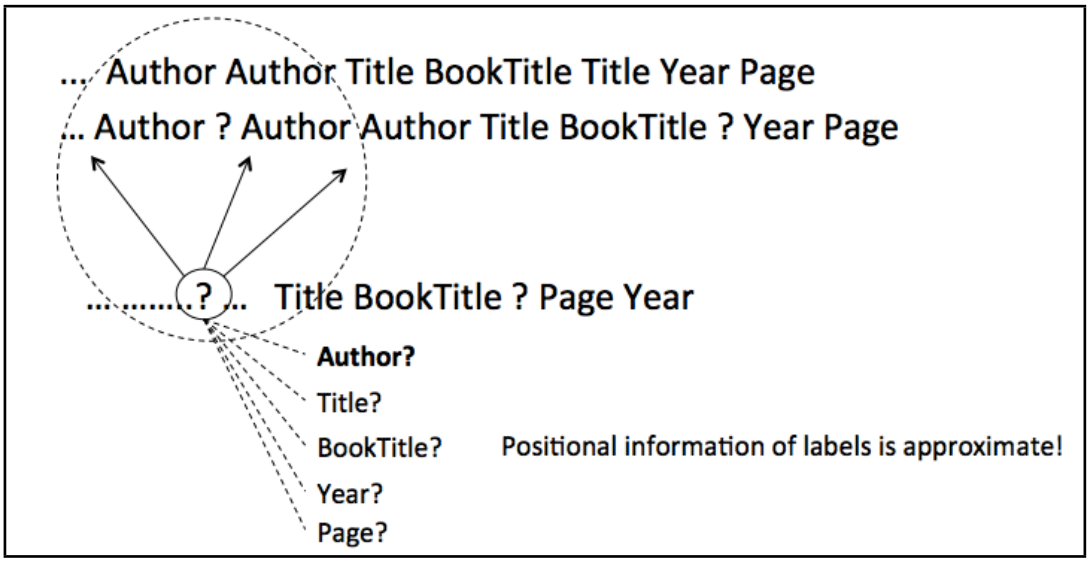


FIGURE 4.2: A demonstration of proximity-based positional model for labels

to obtain labels for text segments in an input list. In the example, when we consider the text segments at the position two in the sequence five in a list, the number of labels "Author" are equal to the number of label "Title". Therefore, if we just consider the number of occurrences in a fixed position, we cannot determine the label of the text segment at the position two.

We argue that the information of a label in a fixed position in different sequences of an input text may not be enough to determine a label of a text segment in the position. Since ones do not know in advance how many text segments there are in each sequence of an input list and the text segments are labeled by matching functions, ones cannot always ensure that majority of correct label always occurs in a fixed position in all sequences of

input lists. Instead, we can consider the occurrences of the labels in different positions in all sequences of the list. That evidence of occurrences of labels near a position should be taken into the probability to have the label in the position. In the example, we can see that the label "Author" occurs frequently in positions around the position two. Therefore, it is high likely that the label of segment at position two is "Author".

In other words, the neighbour positions of a label in an input list should be considered when we compute the probability of the occurrence of the label at a particular position. Based on that idea, we propose a novel *proximity-based positional model for labels* in which we exploit information of different positions of a considering label in the input text to compute the probability to have the label in a particular position.

The key idea of proximity-positional model (PPM) for labels is to define a statistical model for each position of a label in an input list. The PPM for a label at a position would be estimated based on the propagated counts of the label in different positions in the input list. Moreover, the occurrences of a label t at the positions which near a position i will provide more evidence than its occurrences in far positions. In other words, each position of a label will receive propagated counts of the label in near positions. A main technical challenge in the proximity-based positional model for labels in an input list is how to define a propagation function and estimate a positional model for different labels. We will analyse several functions in this chapter and we also show that with some specific choices, our proximity-based positional model for labels covers the fixed-positional model proposed by Cortez et al ([30]) as a special case.

This chapter presents our novel proximity-based positional model for labels to relax a rigid constraint and improve performance of entity extraction by text segmentation. Instead of considering a label in a fixed position in the input text, the distribution of the target label in different positions is taken in our model to measure how likely a label occurs at a given position. The remaining sections of this chapter are organised as follows. Firstly, we introduce and formally define our proposed proximity-based positional model for labels in section 4.2.

Then, we present how to exploit PPM for labels in refinement phase to tackle the problem of entity extraction by text segmentation in section 4.3. After that, our experiments and

evaluations are described and analysed in section 4.4. Eventually, section 4.5 presents some concluding remarks of the chapter.

4.2 Proximity-based positional model for labels

In this section, we present our proposed proximity-based positional model for labels in an input list by considering the distribution of labels in different positions in the list. We firstly formulate the model in section 4.2.1. Then section 4.2.2 investigates some propagation functions which are employed in the model.

4.2.1 Model formulation

As introduced in section 4.1, a proximity-based positional model (PPM) for a label at a position would be estimated based on the propagated label counts from the labels at all other positions in an input list. Specifically, each label at each position of an input list is determined by the evidence of its occurrence to all other positions in the input list and the positions close to the label will get more share of the evidence than those far away. By this way, each position of a label will receive propagated counts of the label from all positions in an input list. From that idea, we formally define a *proximity-based positional model* (PPM) for labels as the following.

Given a list L including n sequences or n lines $L = (l_1, l_2, ...l_n)$, let $T_L = (t_1, t_2, ..., t_N)$ is a list of all possible labels in the list L, N is obviously the number of labels in L.

$c(t, i)$: the number of times the label t occurs at position i in different sequences of the list L.

$k(i, j)$: the discounting factor to the position i from a label at the position j. This factor can be any non-increasing function of $|i - j|$ and called a proximity-based density function. This means that the function $k(i, j)$ returns a high value if the position j is near the position i. In other words, $k(i, j)$ favours positions which are close to i. There are several proximity-based density functions that can be chosen to define $k(i, j)$. Each density function will lead

to a specific PPM for a label. We will explore and analyse different density functions in section 4.2.2.

$c'(t,i)$: the total propagated count of a label t at position i from the occurrences of the label t in all positions in the list L. Formally, $c'(t,i)$ is represented as in the equation 4.1.

$$c'(t,i) = \sum_{j=1}^{N} c(t,j)k(i,j) \tag{4.1}$$

We notice that even if the value of $c(t,i)$ is zero, the value of $c'(t,i)$ may be greater than zero. In other words, $c'(t,i)$ not only considers the positions of the label t at the position i, but also takes into account the neighbour positions of the label t via a proximity-based density function $k(i,j)$.

From the definition of the propagation function $c'(t,i)$ for a label, we have a frequency vector $<c'(t_1,i),\ ...,c'(t_N,i)>$ for different labels at a position i. Accordingly, positional information of each label can be translated to label frequency information in this vector. Based on this formulation, we estimate a proximity-based positional model (PPM) of a label t at position i in a list L as in the equation 4.2.

$$p(t|L,i) = \frac{c'(t,i)}{\sum_{t' \in T_L} c'(t',i)} \tag{4.2}$$

where T_L is a set of labels in L and $c'(t,i)$ is defined by the equation 4.1.

According to the property of the proximity-based propagation function $c'(t,i)$, the value of $p(t|L,i)$ is mainly influenced by labels around the position i, not only a fixed position in the sequences of the list L. In other words, our model can exploit positional information of a label as well as its distribution in a list to incorporate into a statistical modelling framework.

4.2.2 Proximity-based propagation functions for PPM

One of major challenges in PPM for labels is how to define the proximity-based density function $k(i,j)$. As we mentioned in section 4.2.1, the proximity-based density function $k(i,j)$ is a non-increasing function of $|i-j|$ and it favours positions j which are close to i. Therefore, different density functions may lead to different PPMs for labels.

To define such a density function, we follow previous studies about computing the distances between words in a document in information retrieval. Then we consider five different kernel functions $k(i, j)$ in our work and adapt the idea to measure the distances between labels in an input list in our work. Those functions are Gaussian kenel (equation 4.3), Triangle kernel (equation 4.4), Cosine kernel (equation 4.5), Circle kernel (equation 4.6), and Rectangle kernel (equation 4.7) ([39, 86, 66]).

- Gaussian kernel:

$$k(i,j) = exp[\frac{-(i-j)^2}{2\sigma^2}] \tag{4.3}$$

- Triangle kernel:

$$k(i,j) = \begin{cases} 1 - \frac{|i-j|}{\sigma} & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

- Cosine kernel:

$$k(i,j) = \begin{cases} \frac{1}{2}[1 + cos(\frac{|i-j|.\pi}{\sigma})] & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

- Circle kernel:

$$k(i,j) = \begin{cases} \sqrt{1 - (\frac{|i-j|}{\sigma})^2} & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

- Rectangle kernel:

$$k(i,j) = \begin{cases} 1 & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

The examples of the curves of those kernel functions can be illustrated in Figure 4.3. It can be seen in the figure that all the kernel functions have the range values from zero to one and they obtain the highest value when i equals to j. In the kernel functions, the value σ can be viewed as a tuning parameter, which controls the spread of kernel curves, and we can use it to restrict the propagation scope of each label within a web list. The optimal value of σ may vary according to different labels. If a label has wider semantic scope in an input list,

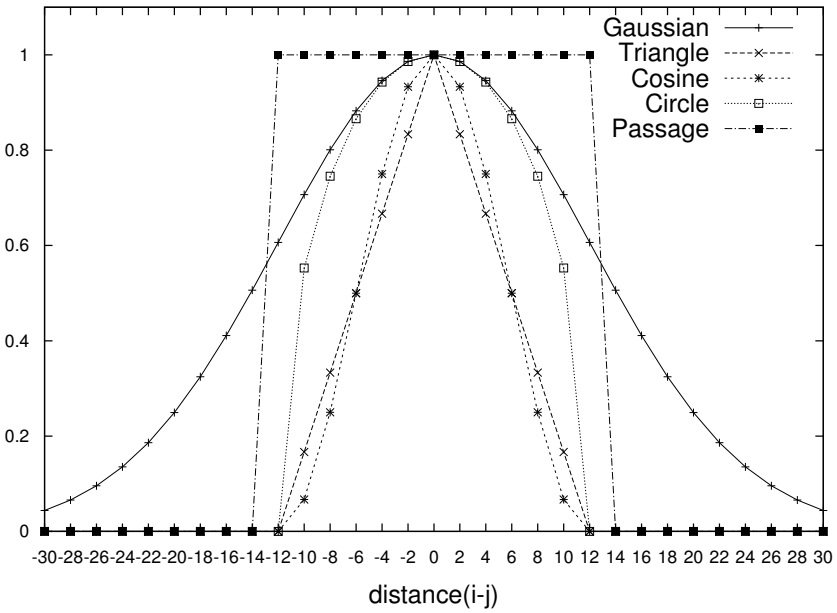


FIGURE 4.3: Proximity-based kernel functions ($\sigma = 12.0$)

the value of σ should be larger. Therefore, based on proximity-based density function, PPM for labels allows us to explore the semantic scope of positions of labels in an input list L.

Moreover, we can prove that the positional model proposed in the study of [30] is actually a special case of our proposed PPM for labels when we set the value of σ to zero. In fact, when the value of σ equals to zero, the expression $|i - j| \leq \sigma$ only returns true when $i = j$. In this case, the proximity-based density function $k(i, j)$ can be represented as in equation 4.8.

$$k(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

From that equation, it can be inferred from the equation 4.1 that the proximity-based propagation function $c'(t, i)$ is always equals to $c(t, i)$ for all values of i. In other words, the value of the proximity-based propagation function $c'(t, i)$ is equal to the number of times the label t occurs at position i in different sequences of the input list L. Therefore, we can conclude that the fixed-positional model in the study of Cortez et al in [30] is a special case of our

proximity-based positional model when we adjust the threshold value σ to zero.

Moreover, we note that the local proximity of labels in a list can be tuned by the parameter σ in our model. If the value of σ is set to a small value in our PPM for labels, we would emphasise on local proximity of labels in a list. Thus, our proposed model can capture the proximity information of labels or their distribution in a list in a statistical modelling framework. Once we obtain a PPM for each position of labels, we use the PPM as a regular model for matching with a label of a text segment at a position in an input list. In section 4.3, we will explain how to employ our proposed model in our self-supervised learning framework to extract information from textual lists.

4.3 Exploiting proximity-based positional model in IETS

The problem of entity extraction by text segmentation has formally defined in section 3.2 of this thesis. In this section, we present the usage of our proposed proximity-based positional model for labels to improve the quality of extracting information of entities from textual web lists.

As presented in section 3.3.1, our extraction methodothology can be described in a sequence of operations which can be grouped into three main phases: splitting phase, matching phase, and refinement phase. In the splitting phase, each string in the input list is split into multiple text segments. In matching phase, we exploit a knowledge base to assign labels for text segments in the input list. In this phase, we exploit a format-based similarity measure to evaluate how likely a text segment should have a label t in knowledge base. After matching phase, some text segments in the input list are unmatched with the knowledge base and therefore they do not have labels. Meanwhile, some other ones may have mismatched labels. These unmatched and mismatched labels will be revised by refinement phase. Due to this phase, they are detected and fixed if they are likely to be incorrect. In our work, our proximity-based positional model for labels proposed in section 4.2 is used to combined with sequential model to revise the results of matching phase.

As described in section 3.3.2, the algorithm of splitting text for each sequence in an

input list is described in Algorithm 1. In our work, we recognise eight primitive datatypes including numbers, date-times, page numbers, volumes and issues, URLs, email addresses, and phone numbers, then define regular expressions which are leant from marks of field values in knowledge base and use them to segment texts in an input list. Due to this process, we can obtain high performance on those simple datatypes and helps achieve effectiveness when we revise the labels of other field values in the input list in refinement phase. In the algorithm of splitting a line into text segments, we initially extract tokens from a string l based on the occurrence of white spaces. For each token t' in a string l, we find a sequence of consecutive tokens starting from t' which satisfies any pre-defined regular expression to group them into a text segment. Moreover, if any two consecutive tokens co-occur in the same instance value according to a knowledge base, they will be in the same text segment. We notice that tokens which do not occur in the knowledge base are always in a single segment.

Given a text segment s, the purpose of label-matching phase is to exploit a knowledge base to assign to s a label t based on the features of the string s. As described in section 3.3.3, we consider the process of of matching a string s and a label t as the process of checking whether the element s is a member of a set with the label t or not. In our work both intensional and extensional definitions are incorporated to label text segments from input lists.

Next, the main purpose of refinement phase is to revise the results of the labelling phase to give labels for unmatched segments and rectify mismatched ones. Similar to the study of Cortez et al ([30]), we also exploited the transitions of labels in an input text to revise the labels. This strategy is based on an assumption that the number of correct labels are more than incorrect ones in an input list. Moreover, it assumes that incorrect labels do not occurs within the same record. Therefore, statistical analysis on labels enables us to detect incorrect ones and fix them. A graphical model is built to represent the likelihood of transitions labels in the input text. For example, if there are several transitions from the label “author” to the label “year”, the probability to revise an unknown label before the label “year” should be higher than probability to have other labels.

As represented in section 3.3.4, a sequential model (SM) for labels in an input list is defined as the following:

- A set of states $T = \{begin, t_1, t_2, ..., t_N, end\}$ where each state t_i describes an entity type labeled to a substring.
- A matrix A where the element a_{ij} is the probability of making a transition from state *i* to state *j*. Each element a_{ij} in the matrix *A* is defined as the equation 4.9.

$$a_{ij} = \frac{Number\ of\ transitions\ from\ state\ t_i\ to\ state\ t_j}{Total\ number\ of\ transitions\ out\ of\ state\ t_i} \tag{4.9}$$

A sequential model is used to revise labels in the results of matching step and helps to improve the recall of extraction results. However, the usage of sequential model may decrease the precision of the system. Therefore, we incorporate our proposed proximity-based positional model with the sequential model to determine the labels of text segments. The proximity-based positional model for labels in an input list can be defined as a matrix P where the entry p_{jk} denotes the probability of the label t_j appearing at the *k-th* position in a sequence of the list. Formally, the value of p_{jk} is defined as in the equation 4.10.

$$p_{jk} = p(t_j|L, k) \tag{4.10}$$

To compute the probability to have a label *t* for a text segment, we combine matching score, sequential model score and proximity-based positional model score by using Bayesian disjunctive operator as in equation 4.11.

$$sim(s,t) = 1 - (1 - f_r(s,t)) \times (1 - f_c(s,t)) \times (1 - f_s(s,t)) \times (1 - f_p(s,t)) \tag{4.11}$$

where $f_r(s,t)$ and $f_c(s,t)$ are accordingly the similarity scores between a text segment s and a label t based on its intensional and extensional definition as in equation 3.7 and 3.8, the value of $f_s(s,t)$ and $f_p(s,t)$ are sequential and positional score which are defined in equation 4.12 and 4.10, respectively.

$$f_s(s,t) = a_{ij} \tag{4.12}$$

$$f_p(s,t) = p_{jk} \tag{4.13}$$

In the equation 4.12, i is the index of the label t in a list of labels T, j is the index of the label of the next segment of s. In the equation 4.13, j is the index of the label t in T and k is the position of s in an input string. The value of a_{ij} and p_{jk} are defined by sequential model and proximity-based positional model as in the equations 4.9 and 4.10, respectively. We note that both matrixes *A* and *P* in both models are built directly by a single pass on the input list L.

4.4 Experiments and results

In this section, we present our experiments to evaluate our method on real datasets to show that our proposed method can achieve better performance than the current state-of-the-art method ([30]). We firstly describe the experimental setup and metrics for evaluations. Then we report experimental results and compare with previous work.

4.4.1 Data settings

We run experiments on the public datasets in two domains: *bibliographic* and *addresses* domain. In each domain, we build a knowledge base and testing data from different data sources. Firstly, in bibliographic domain, we use datasets from experiments in previous studies. They are *CORA* collection ([85]) and *PersonalBib* dataset ([75] [119]). In the domain *Addresses*, we downloaded two datasets *BigBook* and *LARestaurants* from RISE repository ([3]) and then manually label field values in each dataset. Detailed information about the dataset in these domains is summarised in table 4.1.

Domain	Dataset	Attributes	Records
Bibliographic Data	*Cora*	13	500
	PersonalBib	7	395
Address Data	*Bigbook*	5	4,000
	LARestaurants	4	250

TABLE 4.1: Domains and datasets used in our experiments.

4.4.2 Metrics for evaluation

In the experiments, we verify the extraction results in each phase and evaluate how much our proposed techniques can give better performance than techniques used in the existing studies on IETS. In the evaluation, we utilise well-known precision, recall, and F1 measure to compare. The definitions of the measures have been presented in the equation 3.10, 3.11, and 3.12.

4.4.3 Experimental results and evaluation

In this section, we present the experimental results on both domains bibliographic and addresses and compare our results with the current state-of-the-art study proposed by Cortez et al ([30]).

Bibliographic domain

Table 4.2 shows experimental results when we segment citation strings by matching phase only (M), matching and sequential model (M+SM), matching and reenforcement by fixed-positional model (M+PM), matching phase and reenforcement by fixed-positional and sequential model (M+SM+PM), and our proposed technique (M+SM+PPM). We notice that we incorporate format-enhanced matching phase into matching step. Therefore, we can obtain high performance on the quality of information extraction when we perform our method on numeric data types such as page numbers, years, volumes and issues, when compared to

what reported in the study of Cortez et al ([30]). We observe that the extraction process on those data types can be achieved above 94% of F1-measure. Therefore, in the next experiments, we consider how proximity-based positional model can improve the performance on three main labels including "Author", "Title", and "BookTitle".

As we analysed in section 4.2, the kernel function used to estimate the model can determine the performance of each strategy. Therefore, we test our five proposed proximity-based kernel functions mentioned in 4.2.2 and conduct experiments to choose the best kernel function for our PPM for labels. To compare different kernel functions, we systematically vary the values of σ from 0 to 40 in the increments of 0.5 and then observe the changes of the average F1 measure on the extraction of three labels "Author", "Title", and "Booktitle". The results of these experiments are illustrated in Figure 4.4.

Among all the kernel functions, PPM with Gausian kernel gives the best performance when compared to other ones. Moreover, the peak value from the model can be obtained when the value of σ equals to three. The fact that Gausian kernel gives best performance can be explained that the function has a special property: the propagated count drops slowly when the distance value $|i - j|$ is small, but drops quickly as the this distance value is large. This property is reasonable since the dependent labels in an input text often occur around a particular position in different sequences.

Moreover, in order to compare with the baseline and see how PPM can effectively capture proximity of labels in input text, we run experiments by using PPM with Gausian kernel ($\sigma = 3$) and compare with the baseline. The 6^{th} column (M+SM+PPM) in the table 4.2 illustrates the results of our method as compared to the baseline (M+SM+PM) when we set value of σ to three. In general, proximity-based positional model gives better final performance of labelling text segments when compared to fixed-positional model in previous work on bibliographic domain.

Addresses domain

Similar to bibliographic domain, we repeat the experiments in our method and compare with pervious work on Addresses domain. The experiments show that we can obtain 98.11% of

Field	Matching(M)	M+SM	M+PM	M+SM+PM	M+SM+PPM
Author	0.7080	0.7548	0.6774	0.7845	0.7949
Title	0.7882	0.7650	0.6331	0.8217	0.8457
Booktitle	0.7971	0.8609	0.6375	0.7967	0.8068
Pages	0.9961	0.9961	0.9961	0.9961	0.9961
Year	0.9912	0.9912	0.9912	0.9912	0.9912
Volume	0.8483	0.9787	0.7880	0.9404	0.9818
Issue	0.9663	0.9663	0.9663	0.9663	0.9663

TABLE 4.2: Experimental results on *Cora* dataset.

F1-Measure by exploiting format-related features which are implemented in some simple regular expressions to recognise phone numbers. We also vary the values of σ and see how PPM model can improve performance of extraction as compared to the study of Cortez et al ([30]). It is interesting that the best final performance of our method when we use PPM model with different kernels is similar to performance we obtain by using a fixed positional model. That best performance is obtained when we set σ to be zero. As we have proven in the section 4.2.2, when fixed-positional model is actually a case of our model when we set the value of σ to be zero. Therefore, the results in two model in that case are similar.

The results of the performance can be explained by following reasons. Firstly, different from *bibliographic domain*, we notice that the dataset *LARestaurants* in *Addresses* domain is quite regular. Each field value includes only few tokens and the lengths of sequences and field values in the dataset are quite similar. All address strings in the dataset are written in a single order of field values.

Moreover, after utilising a simple application to count the common tokens in different field values within *Bigbook* dataset and *LARestaurants* dataset, we know that there is no overlapping token between any two sets of field values in both the knowledge base and testing data. Therefore, when we perform matching phase, all text segments are assigned to a correct label or an empty label. In addition, the number of tokens in the values of each field

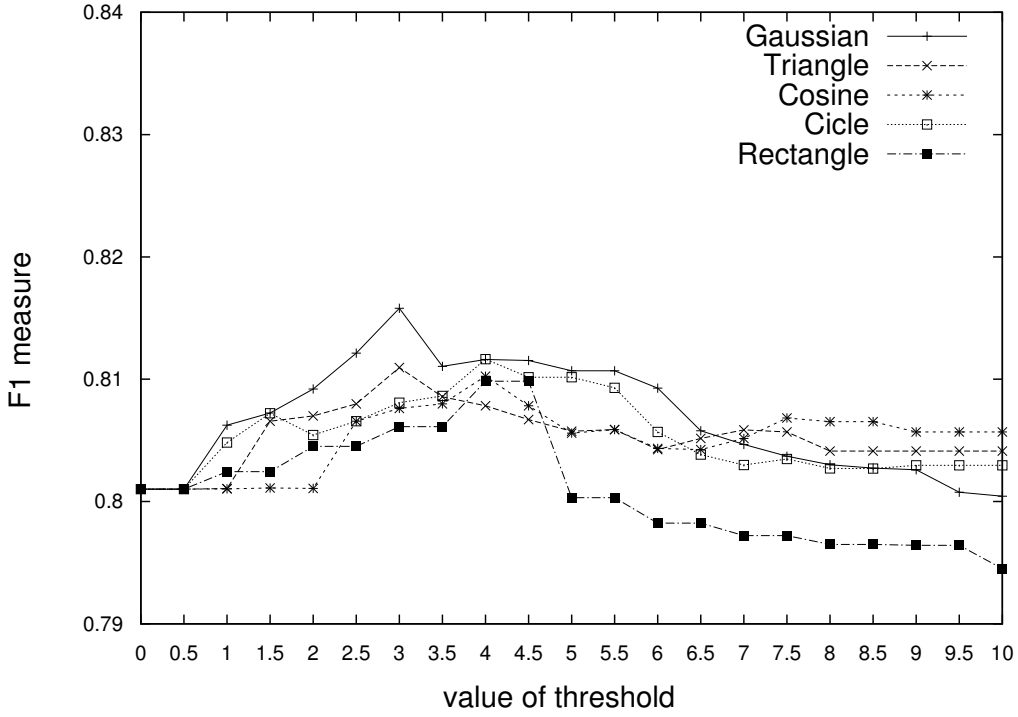


FIGURE 4.4: Performance on Cora dataset with different propagation kernel functions

in testing data are similar. Therefore, the positions of text segments having the same labels are quite similar in different sequences. Meanwhile, citation strings in bibliographic domain can have different authors, different of length of paper title. Therefore, the positions of text segments of a label can be different.

From those experiments, we can conclude that our proposed proximity-based positional model for labels is helpful to capture the distribution of labels in different positions in a list. It can be used in information extraction by text segmentation to deal with the cases in which labels are not always in fixed positions in different sequences of a list.

4.5 Summary

In this chapter, we have presented a novel technique to incorporate positional information of labels in an input list to improve the quality of entity extraction by text segmentation. Our proposed proximity-based positional model for labels considers related positions of labels

Field	M	M+SM	M+PM	M+SM+PM	M+SM+PPM
Name	0.6182	0.8148	0.8903	0.9724	0.9724
Street	0.9073	0.8298	0.8591	0.9808	0.9808
City	0.7388	0.9845	0.8388	0.9857	0.9857
Phone	0.9811	0.9874	0.9817	0.9923	0.9923

TABLE 4.3: Experimental results on *LARestaurants* dataset.

appearing in different sequences of an input list. Therefore, it can capture the distributions of labels in different sequences in an input list as an evidence to the occurrence of the labels in a particular position. We have shown that our proximity-based positional models for labels is more flexible and robust than fixed-positional model in previous study when it can relax the rigid constraint on fixed positions in the previous study as a special case.

Moreover, we have studied five different proximity-based density functions to estimate our proposed proximity-based positional model. Experimental results show that the Gaussian density kernel helps to achieve the best performance and the results of our proposed techniques yield higher quality of information extraction than the state-of-the-art method when we combine it with the sequential model to revise the results of labelling phase in our framework. Eventually, we have partially published the achieved results of this chapter in our publication in [61].

The idea of employing approximately positional information of text segments stimulates us to observe an input list in a vertical view. In that view, the text segments belonging to the same concept or attribute in an input list could be aligned into groups or columns. In the next chapter, we continue to extend our proposed framework to extract information of entities from lists by exploiting the structural similarity between text segments in different sequences of an input list. Similar text segments would be grouped together into clusters or columns before their labels are revised by a graphical model. Due to this clustering process, we could reduce the dependency on knowledge base on labelling phase when building a graphical model in extraction phase. In chapter 5, we will present the ideas in detail.

Chapter 5

Reducing Dependency on Knowledge Base by Structural Similarity

5.1 Introduction

The self-supervised learning framework for information extraction about entities from textual lists proposed in chapter three is robust when it follows an unsupervised approach and does not require manually training data as previous studies. In the framework, the majority of labels which are correctly assigned in matching step can help to build an HMM-based graphical model to assign unmatched labels and rectify incorrect or mismatched ones. In other words, it still exploits an implicit assumption that high overlap between a knowledge base and input lists is required to segment texts and label them before the labels in the matching step are revised by the graphical model.

In this chapter, we present a novel technique to improve the framework when we keep its good features but we reduce the dependency on overlapping terms between knowledge bases and input lists when building its extraction model. We observe that the sequences of a list may contain different number of field values but the field values in sequences are often written in similar formats. An example to illustrate our intuition can be seen in Figure 5.1. The street addresses of restaurants in the example list often starts with a number, then one

Bartlett Pontiff Stewart, 1 Washington St, Glens Falls, NY, 518 792-2117
Bond Schoeneck & King Llp, 111 Washngtn Av, Albany, NY, 518 462 7421
Carl G Dworkin, 44 Bentwood Ct, Albany, NY, (518) 452-5442
Carter Conboy Case Blackmore, 20 Corporate Woods Blvd, Albany, NY, 5184653484
Alan E Goldstein, 1 Calvary Dr, New City, NY, 914 6391771
A A – Lert Locksmith Inc, 1907 Coney Island Ave, Brooklyn, NY, 718 627 1577
Bake's Motorcycle Parts, 1 Bellinger St, Little Falls, NY, 315 823 926

FIGURE 5.1: An example of lists in Address domain

or two tokens and ends with a short abbreviation. Similarly, the values of the field "Phone number" in the list only contain numbers. Some phone numbers or names may contain punctuation marks such as parentheses or a dash in the value. However, those punctuation marks do not often occur in the field values in different sequences.

To reduce the dependency on a knowledge base whilst we label lists, our intuitive idea is to exploit structural similarity between field values in different sequences within an input list to align them into groups or columns before we assign labels to them and revise their labels by using a graphical model. Each group or column contains text segments belonging to the same concepts and then the groups are labelled by using a knowledge base. In order to realise that idea, some challenges need to be addressed and solved as below.

Firstly, whilst the current framework mainly exploits overlapping terms between a knowledge base and input list to perform its segmentation step, our goal is to try to reduce the dependency on a knowledge base. Therefore, we need a technique to improve the segmentation step to generate candidate text segments without using much overlapping terms on knowledge base. We observe that although punctuation marks may occur in field values in a list but some certain punctuation marks are often used as delimiters to separate field values in similar sequences of a list. Therefore, it is necessary to distinguish which punctuation marks are delimiters to separate field values in sequences of an input list. It can be seen in Figure 5.1 that the punctuation mark "&" is not a delimiter and it is a part of the names of restaurants. Moreover, a delimiter that is used to separate a value pair in a position may not

be a delimiter to separate another value pairs in other positions in a list. For example, the punctuation mark "&" is a delimiter to separate author names in a reference but it can be a part of a conference name in a reference string in bibliographical domain. To tackle those challenges in our work, we firstly exploit the style similarity of sequences in a list to improve the segmentation step. We consider the distribution of punctuation marks in the sequences of an input list and we devise a proximity-based positional model for delimiters to detect how likely a punctuation mark should be a delimiter to separate field values in a particular position in the sequences of a list. This model is actually similar to the PPM for labels which has been presented in chapter four. However, we adapt the idea for delimiters in a list instead of labels assigned by a knowledge base. As we presented in our study in [62], the main role of the model is to detect delimiters in a list and use them to perform segmentations.

Secondly, the contents of text segments belonging to the same concept may not be identical although they are written in similar formats. Therefore, current string matching techniques cannot be applied to cluster candidate text segments into groups or columns. It is essential to define a novel similarity model for field values to match the candidate text segments belonging the same concepts or attribute in different sequences into groups or columns. In order to measure the similarity between two text segments in input list, we propose a novel structural similarity measure to evaluate how likely two segments should be aligned into the same group in this chapter. Different from traditional similarity measures which focus on the contents of input strings, our proposed structural similarity measure is defined by exploiting robust features on structures of two text segments within a list ([60] and [62]).

Finally, different sequences of an input list may contain different number of field values. This means that similar text segments may locate in different positions in sequences of a list. In order to group text segments in the list into groups, we need to design an algorithm which allows to group those similar text segments in approximate positions of sequences into group or columns. We devise a data shifting-alignment technique to cluster similar text segments into groups by using the proposed structural similarity measure. Our proposed technique exploits positional information of text segments to combine with structural similarity to discover the repeated patterns among portions of strings within a list to group text segments in

an alignment process. The labels of text segments in the groups are then exploited to build a graphical model and revise the results for information extraction.

The remaining sections of this chapter are organised as follows. Firstly, a proximity-based positional model for delimiters is described in section 5.2. After this model, we respectively present our proposed structural based similarity measures between field values in section 5.3 as well as data shifting-alignment technique in section 5.4. Then, we describe how to incorporate those techniques to extract information of entities in our framework in section 5.5 before experimental results are analysed in section 5.6. Eventually, section 5.7 presents the concluding remarks of this chapter.

5.2 Proximity-based positional model for delimiters

To improve the quality of text segmentation step, we exploit a proximity-based positional model for delimiters in a list. The purpose of this model is to compute how likely a delimiter occurs at a particular position in an input list. Intuitively, if a punctuation mark frequently occurs at a position in a list, it is high possibility that the punctuation mark is a delimiter to separate field values. Similarly, if a text block v is bound by two punctuation marks d_i and d_j and both d_i and d_j accordingly occur frequently at the position i and j in the input list L, it is highly possibility that the delimiters d_i and d_j can be used to separate the text block v. Since the number of field values in each sequence in a list could be different and they may have different tokens or length in an input list L, the delimiters to separate the field values may not be put in fixed positions in different sequences of L. An example to illustrate the issue can be seen in the list in Figure 5.1, the punctuation mark "," (comma), which is used to separate restaurant names and street addresses in the list, is not always located in a fixed position in different sequences. Specifically, it is located after the third token in the first sequence and after the fifth token in the second sequence, and the forth token in the forth sequence. However, because of the format similarity of sequences in a list, they are often written approximately around a particular position in different sequences of an input list.

To model the probability to have a particular delimiter d_i at a position i, we propose a

proximity-based positional model (PPM) for delimiters to measure how likely a delimiter occurs at a position in an input list. Then we use the measure as a score to determine if a punctuation mark is a delimiter to separate field values. The PPM for a delimiter at a position would be estimated based on the propagated counts from the delimiters at all other positions in an input list. The probability of each delimiter at each position of the input text is determined by the evidence of its occurrence to all other positions in the input text and the positions close to the delimiter will get more share of the evidence than those far away. By this way, each position will receive propagated counts of delimiters from all positions in the input text. We formally define a *proximity-based positional model for delimiters* in an input list L as follows.

Given an input list L including n sequences $L = (l_1, l_2, ...l_n)$. We denote $D_L = (d_1, ..., d_i, ..., d_N)$ as a list of all possible delimiters in the list L, where N is obviously the number of delimiters in L.

$c(d, i)$: the number of times a delimiter d occurs at position i in different sequences of the list L.

$k(i, j)$: a discounting factor to position i from a delimiter at position j. This factor can be any non-increasing function of $|i - j|$ and called a proximity-based density function. This means that $k(i, j)$ favours positions close to i. Several proximity-based density functions can be chosen to define $k(i, j)$ and each density function will lead to a specific PPM. We can follow previous studies about computing the distance of words in a document in information retrieval to define a density function for delimiters. An exploration on choosing the best density function for delimiters will be a future study. In our study, we employ a Gaussian kernel function ([86]) to define the density function as in equation 5.1.

$$k(i, j) = exp[\frac{-(i - j)^2}{2\sigma^2}] \tag{5.1}$$

An example of the curve of the kernel function with $\sigma = 5$ is illustrated in Figure 5.2. It can be seen in the figure that the kernel function has a range of values from zero to one and they obtain the highest value when i equals to j.

$c'(d, i)$: the total propagated count of the delimiter d at position i from the occurrences of

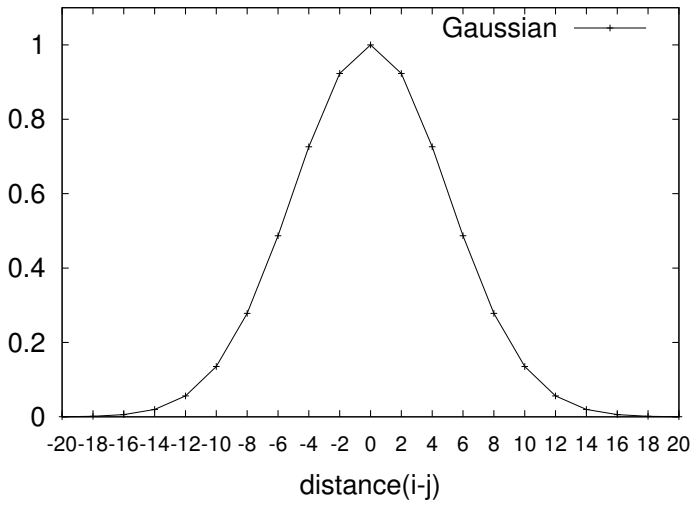


FIGURE 5.2: Gaussian kernel function ($\sigma = 5$)

the delimiter d in all positions in the list L. Formally, $c'(d, i)$ is represented as in the equation 5.2.

$$c'(d, i) = \sum_{j=1}^{N} c(d, j)k(i, j) \tag{5.2}$$

We notice that even if $c(d, i)$ is zero, $c'(d, i)$ may be greater than zero. In other words, $c'(d, i)$ not only considers the positions of a delimiter d at a fixed position i, but also takes into account the neighbour positions of the delimiter d via a proximity-based density function.

In the density function $k(i, j)$, σ is a tuning parameter, which controls the spread of kernel curve to restrict the propagation scope of each delimiter in different sequences of a list. The optimal value of σ may vary according to different delimiters. If a delimiter has wider semantic scope around a position in a list, the value of σ should be larger. Due to the property of a proximity-based density function, a PPM for delimiters allows us to explore the scope of positions of delimiters in a list.

From the propagation function $c'(d, i)$, we have a frequency vector $<c'(d_1, i), ..., c'(d_N, i)>$ for delimiters at a position i. Accordingly, positional information of each delimiter can be translated to frequential information of delimiters in this vector. Based on this formulation, we estimate a *proximity-based positional model of a delimiter* d at a position i in a list L as

in the equation 5.3.

$$p(d|L,i) = \frac{c'(d,i)}{\sum_{d' \in D_L} c'(d',i)} \tag{5.3}$$

where D_L is a set of all possible delimiters in L and $c'(d,i)$ is defined in the equation 5.2.

According to the property of proximity-based propagation function, the value of $p(d|L,i)$ is mainly influenced by delimiters around the position i in the list L. In other words, our model can capture positional information of delimiters in the sequences of a list and incorporate it into a statistical model. Moreover, if the value of σ is set to a small value, we would emphasise on local proximity of delimiters. The balance of local proximity evidence of delimiters in a list can be tuned by the parameter σ. Thus, our proposed model can capture proximity information of delimiters in a statistical modelling framework. Once we obtain a proximity-based positional model for each position of delimiters, we use the model for matching with a delimiter occurring at a position in a list.

5.3 Similarity model for text segments

One of the basic operations in our extraction technique is the ability to put text segments of the same concept into a group or column. As mentioned in section 5.1, it is necessary to define how similar two segments are similar based on their features. In this section, we present our proposed structural similarity to combine with content similarity to measure how likely two text segments are similar.

5.3.1 Structural similarity

This similarity describes the way field values or text segments display in a textual list. We firstly consider the features which provide general information of the strings which form two text segments v_1 and v_2. Those features include: (1) number of letters, (2) percentage of lower case letters, (3) percentage of upper case letters, and (4) percentage of digits in a value. For each feature f_i, we compute a numeric value for the feature and the similarity of

v_1 and v_2 on a feature is defined as in equation 5.4.

$$sim_{f_i}(v_1, v_2) = 1 - \frac{|a_i - b_i|}{max(a_i, b_i)} \tag{5.4}$$

where a_i and b_i are numeric values for a particular feature of values v_1 and v_2.

Beside the general characteristics of strings as above, we also consider the similarity on the organisation of tokens in a string. We observe that two field values or text segments belonging to the same group in an input list often share similar format or representation style. For an instance, person names (e.g "D T Huynh") often start with some abbreviations, which include capital letters, followed by a word which represents a family name. Based on the observation, we devise a format-related similarity of two values v_1 and v_2. We firstly define a set of masks to represent the tokens in the textual string of each value. We tokenise the strings v_1 and v_2 and encode them by using *symbol masks*. For example, the value "D T Huynh" is encoded as "[A-Z] [A-Z] [A-Z][a-z]+", where the mask [A-Z] represents an uppercase letter, the mask [a-z]+ represents a consecutive string of one or many lowercase letter. The similarity between two values v_1 and v_2 is computed based on those sequences of masks. This idea is actually adapted from the study of Borkar et al ([16]) which employs *symbol masks* to capture the format of values in an inner HMM. However, their work utilised a training dataset to build an HMM-based statistical model to capture the format of strings. Meanwhile, our study utilises the concept *symbol masks* to define our structural similarity measure.

Given two values v_1 and v_2, we encode v_1 and v_2 by using the symbol masks as above to obtain two sequences of masks for v_1 and v_2. The distance measure between two masks of v_1 and v_2 is defined as the minimum number of insertions, deletions or substitutions to transfer from this mask to the other one. Then, we apply dynamic programming ([69]) to find the minimum number of operations. Formally, the format-related similarity between two values v_1 and v_2 is illustrated as the equation 5.5.

$$sim_F(v_1, v_2) = 1 - \frac{dist(m_{v_1}, m_{v_2})}{|m_{v_1}| + |m_{v_2}|} \tag{5.5}$$

where m_{v_i} is the sequence of masks encoded for the value v_i, $|m_{v_i}|$ is the number of masks in the sequence, and $dist$ is an edit distance function between two sequences of masks.

As proposed in [60], the structural similarity $sim_S(v_1, v_2)$ is the average of all feature similarities (equation 5.6).

$$sim_S(v_1, v_2) = \sum_{i}^{n} \alpha_i \times sim_{f_i}(v_1, v_2)) \tag{5.6}$$

where n is the number of features of v_1 and v_2 and f_i is a particular feature.

5.3.2 Content similarity

Beside structural similarity, we also consider the similarity of the contents of two text segments. If two text segments or field values share some certain keywords, it will be high possibility that they belong to the same concept and therefore they should be align in the same group. For example, the segments "IEEE Transaction on Knowledge and Data Engineering" and "IEEE Transaction on Multimedia" share some common keywords such as "IEEE" and "Transaction" and both of them are the names of the same concept "journals". Therefore, the content-related similarity measure sim_C can be defined by using a string similarity function between two text segments. A large number of approximately string matching techniques have been proposed in the literature. Popular measures include edit distance functions ([40]), Jaccard coefficient, Cosine similarity measure in information retrieval ([92]), and their extensions to utilise *q-grams* instead of words ([48]). In our work, we adapt the idea of using q-grams in [48] to define the content-related similarity measure. As formulated in equation 5.7, it is the Jaccard similarity between two sets of q-grams of the text values v_1 and v_2.

$$sim_C(v_1, v_2) = \frac{|qg(v_1) \cap qg(v_2)|}{|qg(v_1) \cup qg(v_2)|} \tag{5.7}$$

where $qg(v_i)$ is the set of q-grams associated with the text segment v_i.

5.3.3 Knowledge base support

Intuitively, if two field values v_1 and v_2 co-occur in an attribute of a table or a knowledge base, they should belong to the same group. Since the labels of text segments are assigned in the matching step, two segments in different sequences of an input list with the same

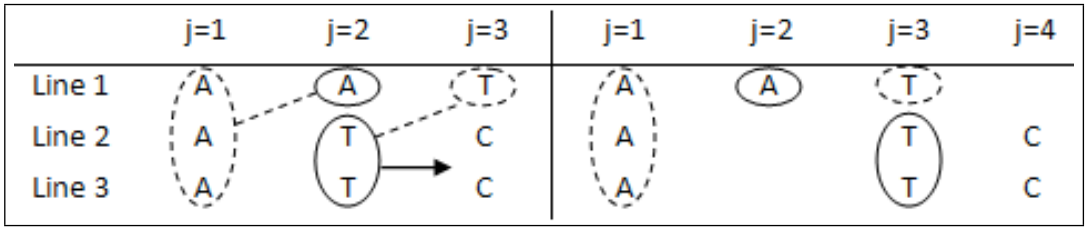


FIGURE 5.3: A demonstration of data shifting-alignment technique

label should be grouped into the same cluster. We set the similarity $sim_K(v_1, v_2)$ between two values v_1 and v_2 to one if they co-occur in an attribute of a knowledge base or zero otherwise.

Given two field values v_1 and v_2, the similarity score between v_1 and v_2 is defined by the weighted sum of the similarity scores of all features between them. We combine them together to define a score function as in the equation 5.8.

$$sim(v_1, v_2) = w_1 * sim_C(v_1, v_2) + w_2 * sim_S(v_1, v_2) + w_3 * sim_K(v_1, v_2) \quad (5.8)$$

where sim_i's are the similarity between v_1 and v_2 on their different features; the weights w_i's are real numbers in [0, 1] and their total is one.

5.4 Data shifting-alignment technique

The purpose of data alignment phase is to align similar text segments in different sequences into groups by using their similarity scores. It can be seen that text segments in the same position in different sequences of a list often belong to the same concept and can be clustered in a group. However, this assumption is not always correct because some field values in a sequence can be missed or the numbers of field values are different in sequences. Therefore, we cannot simply use only positional information of text segments to cluster text segments in an input list into groups.

An example of the problem can be illustrated in Figure 5.3. In the example, each letter *'A'*, *'T'*, and *'C'* accordingly stands for an author name, a paper title and a conference name in bibliography domain. Because there are differences between the number of field values

Algorithm 2: Data alignment

Input: A two-dimensional list of segments in input list L

Output: A set of groups R

```
1  j = 0
2  while true do
3      G[j] = [ ]
4      for i in range(0, |L|) do
5          G[j].append(L[i][j])
6      end
7      if G[j] is empty then
8          break loop
9      end
10     V = CLUSTERING(G[j])
11     c = 0
12     if |V| > 1 then
13         c = CHOOSE-GROUP(V, j, R)
14         for k in range(0, |V|) do
15             if k != c then
16                 for L[i][j] in V[k] do
17                     insert NIL at position j of L[i]
18                 end
19             end
20         end
21     end
22     R.append(V[c])
23     j = j + 1 // move to next position
24 end
25 return R
```

(e.g author names) in different lines of the input list, the text segments in the position two of the sequences are clustered into two groups *'A'* and *'T'*. Therefore, we need to have a method to move the group *'T'* to the next position so that the text segments in the group can be aligned with the text segments in the next positions.

We propose a shifting-alignment technique to overcome this problem in the data alignment phase. We firstly exploit positional information of text segments and their similarity scores to cluster them into groups. If there is only one group returned from the procedure, this means that all text segments will belong to the same group. In contrast, if a list of groups is returned, we keep one group in the list to obtain a group and perform a shifting step to move the remaining groups to the next position and the alignment process will be repeated for next positions to align the text segments in those groups.

Algorithm 2 illustrates steps in our alignment phase. For each column index j, we consider all segments at the index j in all strings of the input list L (lines 3-8). Then we partition them into groups by using our proposed structural similarity (line 12). If there is only one group returned by the clustering algorithm (line 25), we put it into the results (line 22). Otherwise, we will choose a group $v[c]$ in the list V according to the similarity between each group and preceding and succeeding groups (line 14) and then we shift remaining segments in other clusters to next positions. The shifting step is performed by inserting a NIL value into a position in a list (line 17). Finally, we move to the next column index (line 26) and the process is repeated until all segments are aligned into groups (lines 9-11).

In order to cluster similar text segments in a list into groups, we adapt the idea of an agglomerative clustering algorithm ([63]) to cluster text segments into groups by their structural similarity scores. The clustering step (line 10) in the algorithm 2 is illustrated in Algorithm 3. In the algorithm, we adapt the idea of agglomerative clustering algorithm to cluster text segments into groups. Initially, each group of G contains a segment in V (lines 2-4). Then we merge any two groups which return the highest similarity and the similarity is greater than a threshold θ_s (lines 7-21). This process is repeated until we cannot find any two groups whose similarity above the threshold (line 22). After we cluster the text segments, we obtain a set of groups V and each group contains a list of elements of the same concepts.

Algorithm 3: Clustering algorithm for text segments

Input: A list of segments V, a threshold θ_s

Output: A set of groups G

```
G = [ ]
for v in V do
    G.append([v])
end
while |G| > 1 do
    best = 0
    for i in range(0, |G| − 1) do
        for j in range(i + 1, |G|) do
            sim = similarity(G[i], G[j])
            if sim > best then
                best = sim
                a = G[i]
                b = G[j]
            end
        end
    end
    if best > θ_s then
        remove a, b from G
        add a ∪ b into G
    else
        break loop
    end
end
return G
```

Algorithm 4: Choosing a group for data alignment

Input: A list of groups V at the position j; R: preceding groups of V in a list

Output: Index of a chosen group in V

```
S = [ ]
for i in range(0, |L|) do
    for y in range(j + 1, |L[i]|) do
        S.append(L[i][y])
    end
end
score = 0
for k in range(0, |V|) do
    sim_S = similarity(V[k], S)
    sim_P = similarity(V[k], R[j − 1])
    if sim_P − sim_S > score then
        score = sim_P − sim_S
        c = k
    end
end
return c
```

In order to choose a group in a list of groups in the alignment results, we employ some heuristics on the groups in a list. Firstly, given a group V_c in a list of groups, if there is one or many other groups in the next positions which are similar to the group V_c, the group V_c should be moved to the next position so that it can be aligned with the other groups. For example, let's consider the group *'T'* at the position two in Figure 5.3. Because there is a group *'T'* at position three which is similar to the current group *'T'* at position two, the group *'T'* at position two should be moved to the next position so that it can be aligned with the group *'T'* in the next position in the next processing step. Secondly, if a group V_c is similar to a group in the previous position in input list, the group V_c should be kept in the position

Algorithm 5: Text-blocking phase

Input: An input list of sequences L

Output: A list of text segments in the sequences of L

```
for each sequence L[i] ∈ L do
    split L[i] into segments by punctuations
end
build proximity-based positional model P for delimiters in list L
for each sequence L[i] in L do
    for each punctuation d_ij ∈ L[i] do
        if P(d_ij|L, j) < θ_p then
            merge two neighbour text segments of d_ij
        end
    end
end
```

to be aligned into a group and other groups in the list should be moved to new positions. For example, the group *'A'* at position two in Figure 5.3 is similar to the group *'A'* in the previous position. Therefore, we should keep the group *'A'* in alignment results and move the group *'T'* to the next position so that it can be aligned with other text segments.

Based on the observations on a group in a list, we define a score for a group V_c which is the least similar to the following groups and most similar to the preceding groups in a list V. It is computed by the subtraction of the similarity between the group V_c and preceding groups and the similarity between V_c and the succeeding groups. The group V_c with the highest score in the list of groups V will be chosen to form a group and then the remaining groups in the list V will be shifted to the next positions in alignment process. Formally, the choice of group V_c from a list of group V can be described in the equation 5.9.

$$V_c = argmax_{v \in V}(sim_P(v) - sim_S(v)) \tag{5.9}$$

where $sim_P(v)$ and $sim_S(v)$ are respectively the similarity between a group v and preceding

and succeeding groups.

The procedure to choose a group from a list is described in the algorithm 4. Firstly, we select all text segments which follow the position j in the input list L and put them into a list S (lines 1-6). Then, we compute the similarity between each cluster in the list V and the succeeding groups S (line 9) as well as the preceding aligned group (line 11). Since a chosen group should be the most similar to the previous groups and least similar to the succeeding groups, we compute the difference between those two similarity values (line 16) and keep the index of a group in V which yields the largest difference (lines 15-18).

Since the algorithm of the data alignment technique (Algorithm 2) composes of the clustering algorithm for text segments (Algorithm 3) and the algorithm of choosing a group for data alignment (Algorithm 4), we firstly analyse the complexity of Algorithm 3 and 4 before we formulate the complexity of Algorithm 2.

In the Algorithm 3, the loop $while$ (lines 5-23) is performed until we have only one group or we cannot merge groups by using threshold anymore. At each step k in the loop $while$, we have to compute all similarity measures between any two groups in $|V| - k$ clusters to find the highest similarity measure. This involves $O((|V| - k)^2)$ time to choose two groups (lines 7 -16). So, the total complexity across all steps is a sum of k from 0 to $|V| - 1$ of $O((|V| - k)^2)$. Therefore, the complexity of Algorithm 3 in the worst case is $O(|V|^3)$, where $|V|$ is the number of text segments in the input of the algorithm.

Actually, the Algorithm 3 can be improved by using a dynamic programming technique. Initially, we compute the similarities of any two groups in $|V|$ clusters and store in memory. Then we compute the highest similarities of all pairs and store them. All operations takes $O(|V|^2)$ time. For subsequent step k, we just have to look-up the highest similarity measure of a pair of clusters in constant time and update the highest similarity from the new merged cluster to other clusters. This takes $O(|V| - k)$ time. Therefore, the time complexity for all subsequent steps is $O(|V|^2)$. Hence, total time complexity is $O(|V|^2)$. We let the implementation of this idea as well as the experiments on the efficiency of the algorithm be our future study.

In the Algorithm 4, we have to compute the similarity between the text segments of each

group $V[k]$ with other text segments of other groups at each step k. Therefore, the complexity of the algorithm in the worst case is $O(|V|^2)$, where $|V|$ is the number of text segments in the input of the algorithm.

In the Algorithm 2 of data alignment technique, the input of both algorithms 3 and 4 is a set of text segments in at the position j of an input list L. We denote that the input list L is formulated as a two-dimensional list $L(M \times N)$, where M and N are respectively the number of strings and the maximum number of text segments in all strings of the input list L. We need to consider all positions of text segments in the Algorithm 2. At each position j, we perform Algorithm 3 and 4 to cluster text segments and choose a group to keep at the position. Therefore, we spend $O(M^3)$ time for the clustering step by using the Algorithm 3 and $O(M^2)$ time for choosing a group by using the Algorithm 4. In other words, we spend $O(M^3)$ time for both algorithms. Since we need to consider N positions of all text segments in the input list L, the complexity of Algorithm 2 is $O(M^3 * N)$. As we have described above, the Algorithm 3 can be optimised to $O(M^2)$ by using a dynamic programming technique. In that case, the complexity of Algorithm 2 is $O(M^2 * N)$.

5.5 Information extraction steps

5.5.1 Text-blocking and matching

The algorithm 5 describes the steps of text-blocking phase by using our proposed PPM. Firstly we use punctuation to split each string in an input list L into segments (lines 1 - 3). In this step, we also merge any two segments which co-occur in the same field in a knowledge base. Then we utilise information of punctuation marks in the input list to build a PPM for them (line 4). This step is implemented by counting the occurrences of punctuations marks in different positions in the input list and computing the probabilities $p(d|L, i)$ which are defined in the equation 5.3. Finally, the PPM is employed to detect whether a punctuation is a delimiter to separate field values or not (lines 7-9). If the probability to have a delimiter at a position generated by the PPM is less than a threshold θ_p, we merge two neighbour text

segments into one segment (line 8). After this step, all sequences in the input list L are split into text segments. The algorithm 5 requires to traverse all sequences in the input list L and use punctuations to split each sequence into text segments. Therefore, the complexity of the algorithm 5 is $O(M \times N)$ time where M is the number of sequences in L and N is the number of delimiters in the list L.

In the matching step, we reuse the matching score defined in the study of Cortez et al ([30]) to compute the matching scores between a field value and an attribute A_j via a fitness function as in equation 5.10. The fitness scores are computed for all tokens *w* in the query string *s* and the label A_j and it is defined as in the equation 5.11.

$$M(s, A_j) = \frac{\sum_{w \in s} fitness(w, A_j)}{|s|} \tag{5.10}$$

$$fitness(w, A_j) = \frac{freq(w, A_j)}{freq(w)} \times \frac{freq(w, A_j)}{freq_{max}(A_j)} \tag{5.11}$$

where $freq(w, A_j)$ is the number of values of the label A_j containing the token *w*, $freq(w)$ is the total number of instance values in the knowledge base containing the token *w*, and $freq_{max}(A_j)$ is the highest frequency of any token in the instance values of the label A_j.

5.5.2 Data alignment

After text-blocking phase, each sequence in an input list is split into a set of text segments. In data alignment phase, the text segments in different sequences of an input list are aligned into groups according to their similarity scores. We exploit the data shifting-alignment technique which has been presented in section 5.4 to cluster the text segments into groups. Finally, the text segments in the same groups are assigned the same labels and they are used to build a graphical model to revise the results in a final refinement phase.

5.5.3 Refinement

The main purpose of refinement phase is to revise the results of the labelling phase to give labels for unmatched segments and rectify mismatched ones. Cortez et al ([30]) exploited the transitions of labels in an input text to revise the labels. This strategy is based on an

assumption that the number of correct labels are more than incorrect ones in an input list. Moreover, it assumes that incorrect labels do not occur frequently within the same record. Therefore, statistical analysis on labels enables us to detect incorrect ones and fix them. A graphical model is built to represent the likelihood of transitions of labels in an input text. For example, if there are several transitions from the label "author" to the label "year", the probability to revise an unknown label before the label "year" should be higher than probability to have other labels.

In our work, we employ a positional and sequential model (PSM) which was proposed in the study of Cortez et al ([30]) to revise the labels in refinement phase. A PSM is defined by the following three components:

- A set of states $T = \{begin, t_1, t_2, ..., t_N, end\}$ where each state t_i represents a label of a text segment.

- A matrix A where each element a_{ij} is the probability of making a transition from state i to state *j*. Each element a_{ij} in the matrix A is defined as the equation 5.12.

- A matrix P where the entry p_{ik} denotes the probability of the label t_i appearing in the position *k-th* in an input list. Formally, p_{ik} is defined as in the equation 5.13.

$$a_{ij} = \frac{Number\ of\ transitions\ from\ state\ t_i\ to\ state\ t_j}{Total\ number\ of\ transitions\ out\ of\ state\ t_i} \tag{5.12}$$

$$p_{ik} = \frac{Number\ of\ observations\ of\ t_i\ in\ k}{Total\ number\ of\ segments\ in\ k} \tag{5.13}$$

Since the sequences of an input list may contain different number of text segments and labels, the usage of sequential model (SM) helps to improve the recall of extraction results but it could decrease the precision of the system. Therefore, positional model (PM) is combined into the PSM model to revise labels in the results of matching step. To compute the probability to have a label t for a text segment, matching score, sequential and positional model score are combined by using Bayesian disjunctive operator, also known as Nosiy-OR-Gate ([84]), as in equation 5.14.

$$sim(s,t) = 1 - (1 - M(s,t)) \times (1 - a_{ij}) \times (1 - p_{ik}) \tag{5.14}$$

where $M(s,t)$ is a matching score between a segment s and a label t, which is defined as in the equation 5.10; i is the index of the label t in a list of labels T, j is the index of the label of the next segment of s; and k is the position of s in an input sequence. The value of a_{ij} and p_{jk} are accordingly defined by sequential model as in the equation 5.12 and positional model in the equation 5.13. Both matrices A and P are built directly by a single pass on an input list.

5.6 Experiments

In this section, we demonstrate in step-by-step our experiments to evaluate how our proposed method performs well on real datasets in different domains. Then we conduct experiments to show how our proposed method achieves as compared to the current state-of-the-art method. We firstly describe the experimental setup and metrics for evaluations. Then we report empirical results in details in this section.

5.6.1 Experimental setup

In the experiments, our priority is to choose the public datasets which were used in previous work or available on the Internet. In the domain *Addresses*, we utilise *BigBook* dataset from RISE repository ([3]) and then manually label field values in the datasets. This dataset contains 4,000 records of addresses and was employed in the experiments of previous studies ([119], [30]). We use 2,000 records to build a knowledge base and other 2,000 records for testing data. Next, we employ journal references in *PersonalBib* and *Cora* dataset used in previous studies of Cortez et al ([30]) and Peng et al ([85]) as data source and testing dataset in bibliographic data domain. Detailed information about the number of records and attributes in each dataset is described in table 5.1. In our experiments, we set the $\sigma = 3$ and the threshold $\theta_p = 0.4$ for the positional model for delimiters. Moreover, we assign equal weights to the different features in similarity measures and set $q = 5$ and the threshold θ_s = 0.3 for the alignment algorithm.

Domain	Data source	Attributes	Records	Testing dataset	Attributes	Inputs
Addresses	*BigBook*	5	2,000	*BigBook*	5	2,000
Bibliography	*PersonalBib*	7	395	*Cora*	7	150

TABLE 5.1: Domains, data sources, and datasets used in the experiments.

In text-blocking phase, we compute the accuracy of segmentation algorithm to generate correct field values in an input list. It is computed by the percentage of text segments which are correctly split by the algorithm as compared to actual field values within the input list. The accuracy of segmentation algorithm is formulated as in the equation 5.15.

$$Accuracy = \frac{|X_i|}{|Y_i|} \tag{5.15}$$

where X_i is the set of correct field values of a concept obtained from our algorithm and Y_i is a set of actual field values of a concept in an input list.

Moreover, we utilise the well-known precision, recall and F-measure in information extraction to evaluate our experiments. The definitions of the measures have been presented in the equation 3.10, 3.11, and 3.12.

5.6.2 Impact of proximity positional model for delimiters

Figure 5.4a and 5.4b accordingly demonstrate the accuracy of the segmentation phase when we perform our text blocking algorithm (PPM) and the algorithm using only punctuation marks (P). We change the size of testing list on *Cora* and *BigBook* dataset and compare the results obtained when we use both algorithms. In general, PPM-based text blocking technique can help to increase approximate two percent of the accuracy of text segmentation phase as compared to the technique using punctuation marks only. In the case of *Cora* dataset, the accuracy of PPM-based technique slightly increases from 75.3% to 84.0% when we increase the number of input strings from 10 to 150. The accuracies obtained from PPM-based technique are higher than the results from punctuation-based approach, which are

from 73.3% to 80.8% in the same size of input strings. Meanwhile, the accuracy of PPM-based technique is quite stable on *BigBook* dataset. Its accuracy is approximate 97.5% on different sizes of input data when compared to 95.0% which is the accuracy obtained by the punctuation-based technique. We note that *BigBook* dataset is quite regular as compared to *Cora* dataset. With small amount of input strings, PPM model can obtain enough statistical information about punctuation for delimiters to perform segmentation step. Therefore, the results on *BigBook* dataset are quite stable as compared to the results on *Cora* dataset.

In the next experiments, we illustrate the impact of the PPM-based segmentation step on the different phases of information extraction process. As described in Figures 5.5 and 5.6, PPM helps to increase the performance of the segmentation step and the results help to improve performance of matching, alignment, and refinement phase. It can be seen from the figures 5.5a and 5.6a that the improvement of performance between P-matching technique and PPM-matching technique is large when the overlap between knowledge base and input list is small and this gap is smaller and smaller when we increase the number of overlapping terms. This shows that when the number of overlapping terms between knowledge base and input list is large, there is not much difference between PPM-matching and P-matching technique. However, PPM-matching technique can help to reduce necessary overlapping terms to obtain a certain performance. This can be explained that when the number of correct segments are high, we may need less overlapping terms to label text segments in an input list. As a result, the performance of alignment and refinement phase can be improved based on the matching results. Moreover, it can be observed that although refinement phase plays important role when text segments are not segmented correctly, PPM still can help to improve the performance of extraction when compared to the punctuation-based segmentation technique.

5.6.3 Impact of previously known data

In this section, we analyse the impact of the overlapping terms between a knowledge base and an input list to illustrate the advantages of our proposed method as compared to ONDUX,

the state-of-the-art method for information extraction by text segmentation. The experiments are performed on both *Cora* and *BigBook* dataset. In the experiments, we vary the number of shared terms in a knowledge base which overlap with an input list and see the quality of information extraction on the input list. We repeat the experiments five times and compute the average F1-measure of all field values in each step.

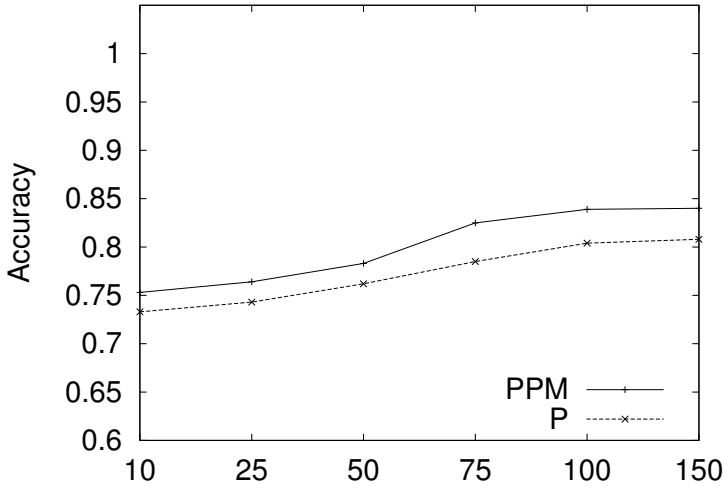
Figure 5.5c and 5.6c accordingly represent experimental results when we vary the number of shared terms between knowledge base and an input list from 50 to 300 terms in *Cora* dataset and from 50 to 1,000 terms in *BibBook* dataset. In general, performance of both methods is higher and higher when we increase the number of shared terms. Especially when the number of shared terms approximately approaches the maximum values, both methods reach similar extraction quality. However, when the number of overlapping terms is not large enough, the performance of ONDUX drops dramatically whilst our method still gives better performance as compared to ONDUX. It can be observed in Figure 5.6c that the F-measure values obtained by ONDUX are quite low when the number of common terms is less than 200. In other words, ONDUX is quite dependent on the overlapping between the knowledge base and the input list. Meanwhile, our proposed method still keeps good performance with more than 79% of F1-measure.

Those experimental results are expected because ONDUX only exploits the overlapping terms to obtain the statistics about the structure of the testing list. Once overlapping terms are not large enough, ONDUX cannot build a good statistical model to revise the results which were generated by their matching step. Meanwhile, our proposed method can exploit the structural information of text segments within a list to cluster them into groups. Then, the group can be labelled with only some overlapping data with knowledge base. As a result, the alignment step in our method can help to increase the number of assigned labels to obtain statistical information about the structure of the input list to revise the results in refinement phase. In practice, the requirements of high overlapping between a knowledge base and input lists could not be easily obtained all the times, especially when we perform an extraction on an arbitrary list. Therefore, we can conclude that our method is more robust than ONDUX in terms of less dependency on the overlapping between knowledge base and input lists.

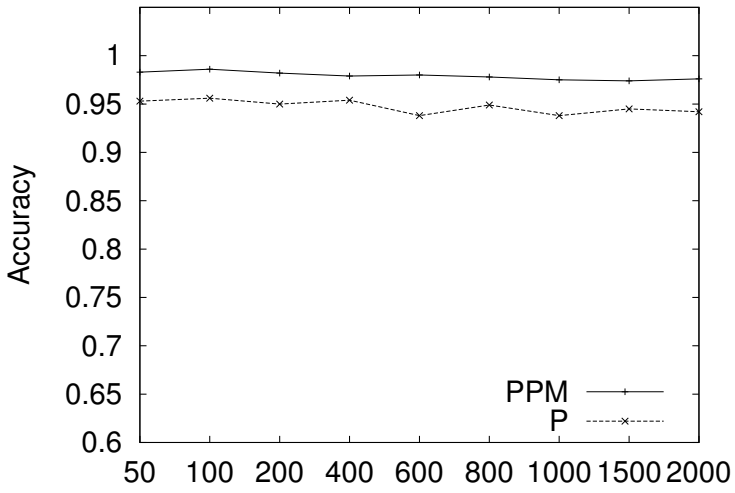
5.7 Summary

In this chapter, we have presented our approach to reduce the dependency on knowledge base for information extraction by text segmentation from textual lists. In our approach, a proximity-based positional model for delimiters is proposed to improve the quality of segmentation steps and the process of information extraction from list. Our model can capture the proximity information of each position of delimiters in a list to obtain statistical information about them and measure how likely a punctuation mark should be a delimiter to separate field values in a list.

Moreover, structural similarity between text segments and sequences is exploited to group similar text segments in different sequences into clusters before we revise their labels by an HMM-based graphical model. We propose a structural similarity to measure how likely two text segments are similar and combine it in a shifting-alignment technique, in which positional information of text segments is combined with a shifting technique to cluster data into groups. We have conducted an extensively experimental study with real datasets in different domains on the web. The experimental results show that our extraction method is robust and can extract information from lists with high performance and less dependent on knowledge base than ONDUX ([30]), the current state-of-the-art study on similar problem of information extraction by text segmentation. Eventually, the results of this chapter have been presented in our publications in [60] and [62].

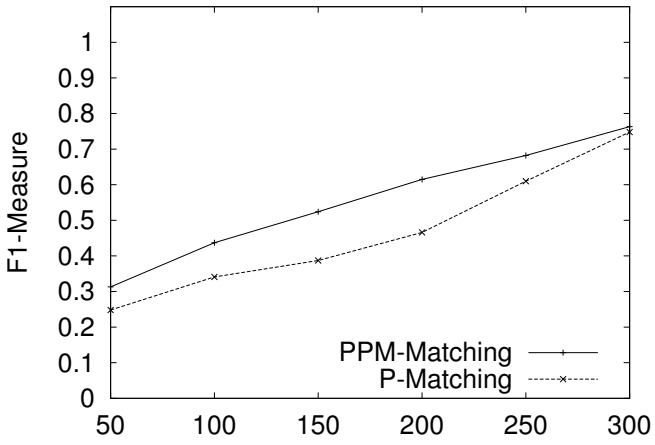


(a) Cora dataset

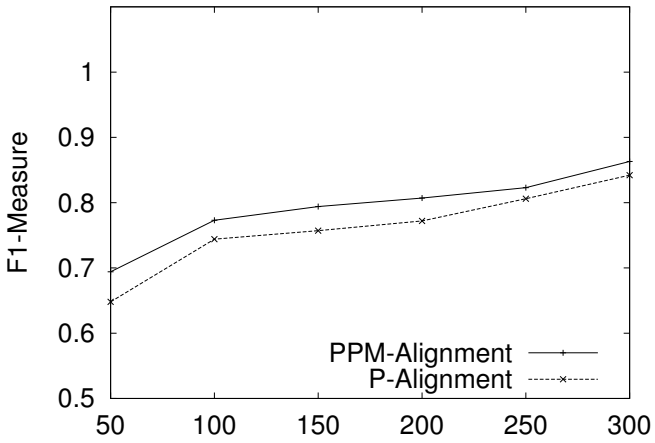


(b) BigBook dataset

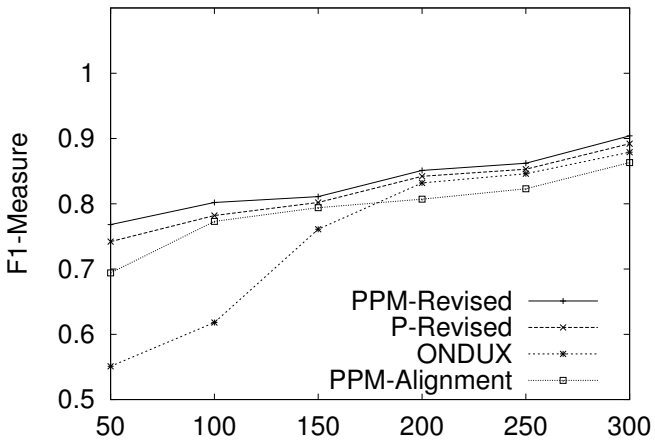
FIGURE 5.4: Accuracy of segmentation phase when varying the size of input list on *BigBook* and *Cora* dataset



(a) Matching phase

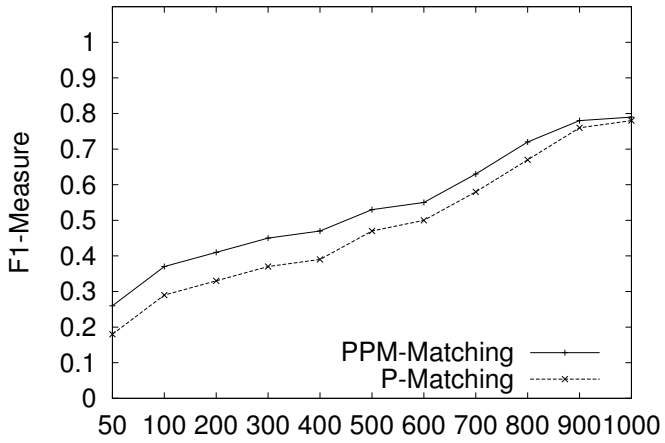


(b) Alignment phase

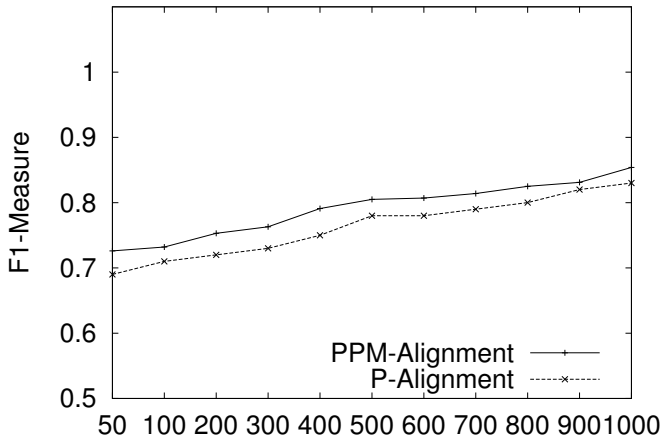


(c) Refinement phase

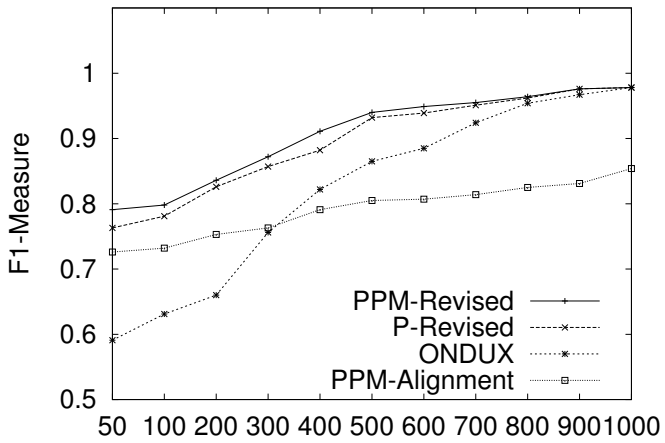
FIGURE 5.5: Performance obtained when varying shared terms on *Cora* dataset



(a) Matching phase



(b) Alignment phase



(c) Refinement phase

FIGURE 5.6: Performance obtained when varying shared terms on *BigBook* dataset

Chapter 6

Epilogue

This chapter presents the conclusions of this thesis. Section 6.1 reviews the research problems and summarises the findings and major contributions from this research project. Section 6.2 discusses and proposes some possible improvements and future studies from the current research results.

6.1 Conclusions

In one sentence to summarise our work, this thesis investigates and devises novel techniques to extract information about entities from textual lists on the web. The primary methodology used in our approach is to employ an existing knowledge base to build a statistical extraction model automatically and then exploit the structural similarity of sequences of input lists to improve the performance of information extraction results. We have proposed a self-supervising learning framework for extraction of information about entities from textual lists which comprises of three phases: a text segmentation phase, a matching phase to assign labels by employing a knowledge base, and a refinement phase to revise the unmatched and mismatched labels by a statistical extraction model.

To improve the quality of the text segmentation and label matching phase, we identify and propose a dyadic representation of membership relations between a text segment and an attribute of data concept. By viewing the problem of labelling as a problem of membership

checking in set theory, we can incorporate both extensional and intensional representation of the relation when we perform the matching phase. The extensional relation between a text segment and an attribute or a label is defined by the similarity between the content of the text segment and all members of the attribute in a knowledge base. Alternatively, the intentional relation between a field value and an attribute is described by specifying a set of lexical-syntactic patterns to identify the field value. Based on those representations, we have developed a format-enhanced labelling technique for the label matching phase in our proposed framework. As presented in chapter 3, our experiments showed that the proposed technique can help to improve the quality of the matching phase and overall extraction results.

Moreover, it can be observed that the labels of the identical attribute in different sequences of an input list are often located in similar positions. We exploit that positional information to improve the quality of entity extraction by text segmentation. To capture the positional information of labels in an input list, we devise a novel proximity-based positional model for labels which considers the related positions of labels which appear in different sequences of input list. Our proposed model captures the distributions of labels in different sequences in an input list as an evidence of the occurrence of the labels in a particular position. As proven in chapter 4, our proximity-based positional models for labels can relax the rigid constraint on fixed positions in the current state-of-the-art study as a special case. The proposed model for labels provides a statistical information on the positions of labels and it is combined with the transitions between labels in an input list to improve the quality of information extraction from the list.

Eventually, instead of depending on the highly overlapping terms between knowledge bases and input lists to label text segments, we exploit structural similarity between text segments in different sequences of an input list to cluster them into groups or columns before we revise the labels in a refinement phase. We argue that the requirements of high overlap between knowledge bases and input lists could not easily be obtained routinely, especially when performing an extraction on an arbitrary list. Firstly, we exploit the style similarity of sequences in a list to perform segmentation. We observe that although delimiters may

occur in field values in a list, certain delimiters are often used to separate field values in the list. From that observation, we adapt the idea of the proximity-based positional model for labels to delimiters to detect how likely a punctuation mark should be a delimiter to separate field values in sequences. The main role of the model is to detect delimiters in a list and use them to perform segmentation. Then, we propose a novel structural similarity measure to evaluate how likely it is that two text segments should be aligned into the same group or column. The proposed structural similarity measure is defined by mainly exploiting robust features on structures of any two text segments within a list. Then we devise a data shifting-alignment technique to group similar text segments into clusters or groups by using the structural similarity measure. Our proposed technique exploits the positional information relating to text segments and combines with this structural similarity to elucidate the repeating patterns among portions of strings within a list in order to group text segments in an alignment process. The labels of text segments in the groups are then exploited to build a graphical model and revise the results for extraction of information. We conduct an extensive experimental study with real datasets on the web. The experimental results show that our extraction method is robust, and performs well with high performance and it is less dependent on knowledge base than the current state-of-the-art study.

6.2 Suggestions for future research

Based on the current studies and research results achieved from this thesis, we suggest some new research topics for future investigation as follows.

1. Currently, the graphical model in the refinement phase in our framework is used to capture the structure of sequences in a list from statistical information relating to labels in the matching phase. In practice, people may include several types of entities in a list. Moreover, information about entities may be written in different formats in a list. An example of this issue can be considered in a bibliographical domain. Scientists may mix the references of their journal publications, technical reports, and book chapters on their home pages and they could be written in different formats, which define different orders of field values, e.g. [Title,

Authors, Conference, Year] or [Authors, Year, Title, Conference]. Although the graphical model in our framework can capture different transitions of labels, it would be more effective to categorise such heterogeneous lists into homogeneous lists before our proposed data shifting-alignment technique is applied. This would require a structured based categorisation technique to operate on such lists to perform entity extraction. This is one of the possible directions for future work.

2. Our proposed structural similarity and data-shifting alignment technique can help to reduce the dependency on a knowledge base when we extract information about entities in lists. However, the current approach still makes an assumption that the attributes of the values to be extracted from an input list are available in a knowledge base when building a graphical model for revising labelling results. In practice, some attributes may be missed in the knowledge base and we do not always have data about all of the types of entities. For example, DBLP ([1]) provides a great knowledge base on references in computer science of bibliographical domain but it does not contain information about the locations of the conferences. Therefore, how to build a statistical extraction model for the problem of entity extraction from lists with some unknown labels would become a challenging and difficult research problem in such situations. Structural similarity and positional information of text segments in a list can be exploited to group or align them into columns if they are represented in similar styles in an input list. Therefore, our proposed techniques in this thesis can be extended to open an opportunity to extract information from any input lists without the assumption. That is one of the future studies that could be undertaken.

3. Information about entities may not be complete in an individual list and it may be scattered in several lists on the web. For example, information about publications of scientists can be found in the list of publications on their home pages but information about the locations of the conferences or journals, their rankings, or reviewers can be contained in other lists on the web. Therefore, a further research problem on entity extraction from lists is how to synthesise information about entities and produce derived tables from such raw lists on the web. That is referred to as the problem of "list stitching" which involves combining lists of entities into a meaningful table and identifying additional attributes and values for its

rows from lists. Those research results would also provide a reasoning capability to detect new relationships between entities from different lists on the web and open an opportunity to study an ecosystem of implicit structured data from lists on the web.

References

[1] The dblp computer science bibliography, available at http://dblp.uni-trier.de/xml/, last visited: April 2014.

[2] Internet live stats - internet usage and social media statistics, available at http://www.internetlivestats.com, last visited: June 2014.

[3] Rise – a repository of online information sources used in information extraction tasks, 1998, available at http://www.isi.edu/info-agents/rise/index.html, last visited: April 2014.

[4] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 20–29, 2004.

[5] E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, and A. Voskoboynik. Snowball: a prototype system for extracting relations from large text collections. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, SIGMOD '01, New York, NY, USA, 2001. ACM.

[6] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proceedings of the 32nd International Conference on Very Large Databases*, VLDB '06, pages 918–929. VLDB Endowment, 2006.

[7] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 337–348, 2003.

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[10] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 119–128, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[11] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 57–64, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[12] T. Berners-Lee. Information management: A proposal, w3c, available at http://www.w3.org/history/1989/proposal.html, last visited: June 2014.

[13] D. M. Bikel, R. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231, February 1999.

[14] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18:16–23, September 2003.

[15] D. Bollegala, Y. Matsuo, and M. Ishizuka. Relational duality: unsupervised extraction of semantic relations between entities on the web. In *Proceedings of the 19th International Conference on World Wide Web*, pages 151–160. ACM, 2010.

[16] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 175–186, 2001.

[17] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB '98: Selected papers from the International Workshop on The World Wide Web and Databases*, pages 172–183, London, UK, 1998. Springer-Verlag.

[18] R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[19] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data integration for the relational web. *Proceedings of the VLDB Endowment (PVLDB)*, 2:1090–1101, August 2009.

[20] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment (PVLDB)*, 1:538–549, August 2008.

[21] M. J. Cafarella, J. Madhavan, and A. Halevy. Web-scale extraction of structured data. *ACM SIGMOD Record*, 37(4):55–61, 2008.

[22] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[23] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filter for approximate membership checking. In *Proceedings of the 2008 ACM SIGMOD International*

Conference on Management of Data, SIGMOD '08, pages 805–818, New York, NY, USA, 2008. ACM.

[24] A. Chandel, P. C. Nagesh, and S. Sarawagi. Efficient batch top-k search for dictionary-based entity recognition. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 28–37, Washington, DC, USA, 2006. IEEE Computer Society.

[25] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 5–16, Washington, DC, USA, 2006. IEEE Computer Society.

[26] S. Chaudhuri, V. Ganti, and D. Xin. Mining document collections to facilitate accurate approximate entity matching. *Proceedings of the Very Large Database Endowment (PVLDB)*, 2(1):395–406, Aug. 2009.

[27] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78, 2003.

[28] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 201–212, New York, NY, USA, 1998. ACM.

[29] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[30] E. Cortez, A. S. da Silva, M. A. Gonçalves, and E. S. de Moura. Ondux: On-demand unsupervised learning for information extraction. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 807–818, New York, NY, USA, 2010. ACM.

[31] E. Cortez, A. S. da Silva, M. A. Gonçalves, F. Mesquita, and E. S. de Moura. A flexible approach for extracting metadata from bibliographic citations. *Journal of the American Society for Information Science and Technology*, 60:1144–1158, 2009.

[32] E. Cortez, D. Oliveira, A. S. da Silva, E. S. de Moura, and A. H. Laender. Joint unsupervised structure discovery and information extraction. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 541–552, New York, USA, 2011. ACM.

[33] I. G. Councill, C. L. Giles, and M. yen Kan. Parscit: An open-source crf reference string parsing package. In *International Language Resources and Evaluation*. European Language Resources Association, 2008.

[34] V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *Journal of the ACM (JACM)*, 51:731–779, September 2004.

[35] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Databases*, pages 109–118, 2001.

[36] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[37] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[38] M.-Y. Day, R. T.-H. Tsai, C.-L. Sung, C.-C. Hsieh, C.-W. Lee, S.-H. Wu, K.-P. Wu, C.-S. Ong, and W.-L. Hsu. Reference metadata extraction using a hierarchical knowledge representation framework. *Decision Support System*, 43:152–167, February 2007.

[39] O. De Kretser and A. Moffat. Effective document presentation with a locality-based similarity heuristic. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 113–120, New York, NY, USA, 1999. ACM.

[40] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19:1–16, January 2007.

[41] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[42] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165:91–134, June 2005.

[43] D. Freitag and A. McCallum. Information extraction with hmm structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 584–589. AAAI Press, 2000.

[44] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Special interest tracks and posters of the 14th International Conference on World Wide Web*, WWW '05, pages 830–839, New York, NY, USA, 2005. ACM.

[45] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The lixto data extraction project: back and forth between theory and practice. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '04, pages 1–12, New York, NY, USA, 2004. ACM.

[46] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava. Using q-grams in a dbms for approximate string processing. *IEEE Data Engineering Bulletin*, 24:28–34, 2001.

[47] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *Proceedings of the 27th International Conference on Very Large Databases*, VLDB '01, pages 491–500, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[48] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an rdbms for web data integration. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 90–101, New York, NY, USA, 2003. ACM.

[49] R. Grishman and B. Sundheim. Message understanding conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[50] P. Gulhane, R. Rastogi, S. H. Sengamedu, and A. Tengli. Exploiting content redundancy for web information extraction. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1105–1106, New York, NY, USA, 2010. ACM.

[51] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274, New York, NY, USA, 2009. ACM.

[52] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment (PVLDB)*, 2:289–300, August 2009.

[53] H. Han, C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox. Automatic document metadata extraction using support vector machines. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital libraries*, pages 37–48, 2003.

[54] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[55] D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, ACL '90, pages 268–275, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.

[56] J. R. Hobbs, J. Bear, D. Israel, and M. Tyson. Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*, pages 1172–1178, 1993.

[57] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[58] W. Hua, T. D. Huynh, S. Hosseini, J. Lu, and X. Zhou. Information extraction from microblogs: A survey. *International Journal of Software and Informatics*, 6:495–522, 2013.

[59] T. D. Huynh and W. Hua. Self-supervised learning approach for extracting citation information on the web. In *The 14th Asia-Pacific Web Conference (APWeb 2012)*, 2012.

[60] T. D. Huynh, J. Xu, S. Sadiq, and X. Zhou. Exploiting structural similarity for automatic information extraction from lists. In *Proceedings of the 14th International Conference on Web Information System Engineering (WISE 2013)*, 2013.

[61] T. D. Huynh and X. Zhou. Exploiting a proximity-based positional model to improve the quality of information extraction by text segmentation. In *The 24th Australasian Database Conference (ADC 2013)*, 2013.

[62] T. D. Huynh and X. Zhou. Automatic information extraction from lists on the web. *Submitted to Journal of the Association for Information Systems*, 2014.

[63] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Survey*, 31(3):264–323, 1999.

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

[65] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The yago-naga approach to knowledge discovery. *ACM SIGMOD Record*, 37(4):41–47, 2008.

[66] M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, Feb. 2001.

[67] N. Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, April 2000.

[68] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

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

[70] G. Li, D. Deng, and J. Feng. Faerie: efficient filtering algorithms for approximate dictionary-based entity extraction. In *Proceedings of the 2011 International Conference on Management of Data*, SIGMOD '11, pages 529–540, New York, NY, USA, 2011. ACM.

[71] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of VLDB Endowment (PVLDB)*, 3:1338–1347, September 2010.

[72] J. Lu, J. Han, and X. Meng. Efficient algorithms for approximate member extraction using signature-based inverted lists. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 315–324, New York, NY, USA, 2009. ACM.

[73] C. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[74] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[75] I. R. Mansuri and S. Sarawagi. Integrating unstructured data into relational databases. In *Proceedings of the 22nd International Conference on Data Engineering*, pages 29–40, 2006.

[76] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[77] A. K. McCallum. Mallet: A machine learning for language toolkit, 2002.

[78] F. Mesquita, A. S. da Silva, E. S. de Moura, P. Calado, and A. H. F. Laender. Labrador: Efficiently publishing relational databases on the web by using keyword-based query interfaces. *Information Processing & Management*, 43(4):983–1004, July 2007.

[79] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings ECML*, pages 318–329, 2006.

[80] K.-R. Mller, S. Mika, G. Rtsch, S. Tsuda, and B. Schlkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, 2001.

[81] N. Nakashole, M. Theobald, and G. Weikum. Find your advisor: Robust knowledge gathering from the web. In *WebDB*, 2010.

[82] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[83] T.-V. T. Nguyen, A. Moschitti, and G. Riccardi. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[84] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[85] F. Peng and A. McCallum. Information extraction from research papers using conditional random fields. *Information Processing and Management*, 42:963–979, July 2006.

[86] D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In *Proceedings of the sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 731–740, New York, NY, USA, 2007. ACM.

[87] F. Reichartz, H. Korte, and G. Paass. Composite kernels for relation extraction. In *ACL-IJCNLP '09: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 365–368, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[88] F. Reichartz, H. Korte, and G. Paass. Semantic relation extraction with kernels over typed dependency trees. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 773–782, New York, NY, USA, 2010. ACM.

[89] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. An algebraic approach to rule-based information extraction. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 933–942, Washington, DC, USA, 2008. IEEE Computer Society.

[90] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1044–1049, 1996.

[91] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data Knowledge Engineering*, 36:283–316, March 2001.

[92] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[93] S. Sarawagi. The crf project: A java implementation, 2004, available at http://crf.sourceforge.net, last visited: April 2014.

[94] S. Sarawagi. Information extraction. *Foundation and Trends in Databases*, 1(3):261–377, 2008.

[95] K. Seymore, A. Mccallum, and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[96] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd International Conference on Very Large Databases*, VLDB '07, pages 1033–1044. VLDB Endowment, 2007.

[97] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[98] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal Molecular Biology*, 147:195–197, 1981.

[99] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272, February 1999.

[100] F. M. Suchanek, G. Ifrim, and G. Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 712–717, New York, NY, USA, 2006. ACM.

[101] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge unifying wordnet and wikipedia. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[102] F. M. Suchanek, M. Sozio, and G. Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 631–640, New York, NY, USA, 2009. ACM.

[103] E. Sutinen and J. Tarhio. On using q-gram locations in approximate string matching. In *Proceedings of the Third Annual European Symposium on Algorithms*, ESA '95, pages 327–340, London, UK, 1995. Springer-Verlag.

[104] Taku. Crf++: Yet another crf toolkit, 2005, available at http://crfpp.sourceforge.net, last visited: April 2014.

[105] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computing Science*, 92:191–211, January 1992.

[106] J. Ullmann. A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. *The Computer Journal*, 20(2):141, 1977.

[107] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[108] P. Venetis, A. Halve, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment (PVLDB)*, 2011.

[109] R. C. Wang and W. W. Cohen. Language-independent set expansion of named entities using the web. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 342–350, Washington, DC, USA, 2007. IEEE Computer Society.

[110] R. C. Wang and W. W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1503–1512, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[111] W. Wang, C. Xiao, X. Lin, and C. Zhang. Efficient approximate entity extraction with edit distance constraints. In *Proceedings of the 35th SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 759–770, New York, NY, USA, 2009. ACM.

[112] D. S. Weld, R. Hoffmann, and F. Wu. Using wikipedia to bootstrap open information extraction. *ACM SIGMOD Record*, 37(4):62–68, 2008.

[113] W. E. Winkler and Y. Thibaudeau. An application of the fellegi-sunter model of record linkage to the 1990 u.s. decennial census. In *Technical report statistical research report series RR91/09, US Bureau of the Census, Washington, D.C.*, 1991.

[114] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 41–50, New York, NY, USA, 2007. ACM.

[115] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 635–644, New York, NY, USA, 2008. ACM.

[116] F. Wu and D. S. Weld. Open information extraction using wikipedia. In *ACL '10: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Morristown, NJ, USA, 2010. Association for Computational Linguistics.

[117] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, NAACL-Demonstrations '07, pages 25–26, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[118] M. Zhang, G. Zhou, and A. Aw. Exploring syntactic structured features over parse trees for relation extraction using kernel methods. *Information Processing & Management*, 44:687–701, March 2008.

[119] C. Zhao, J. Mahmud, and I. V. Ramakrishnan. Exploiting structured reference data for unsupervised text segmentation with conditional random fields. In *Proceedings of the SIAM International Conference on Data Mining*, pages 420–431, 2008.

[120] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW '09: Proceedings of the 18th International Conference on World Wide Web*, pages 101–110, New York, NY, USA, 2009. ACM.