Mining Definition of Term in Scientific Articles

Thesis · July 2013

1 author:

Yiping Jin
Knorex Pte. Ltd.
3 PUBLICATIONS **8** CITATIONS

Honours Year Project Report

Mining Definition of Term in Scientific Articles

By

Jin Yiping

Department of Computer Science

School of Computing

National University of Singapore

2012/13

Honours Year Project Report

Mining Definition of Term in Scientific Articles

By

Jin Yiping

Department of Computer Science

School of Computing

National University of Singapore

2012/13

Project No: H079850
Supervisor: A/P Kan Min-Yen
Advisor: Ng Jun Ping, He Xiangnan
Deliverables:
Report: 1 Volume
Source Code: 1 DVD

Abstract

We consider the identification, demarcation and extraction of definitions present in scholarly documents. Unlike previous approaches, we deem the task of definition extraction as sequence labelling task, adopting state-of-the-art of conditional random fields machine learning methodology to improve system performance. Our implemented definition extraction system, DefMiner, represents the state-of-the-art in definition extraction, incorporating features that exploit different levels of natural language processing. Compared with previous published work, our system improves performance as judged by F_1 significantly, by 12%, achieving an F_1 of 85% on Wikipedia corpus.

We exploit DefMiner to process the sizable *ACL Anthology Reference Corpus* (ACL ARC) – a real-world and large-scale digital library of scientific articles in computational linguistics. The resultant large-scale, automatically acquired glossary of terms, represents the combined terminology defined over several thousand individual research articles. Analyzing the glossary, we uncover interesting distributional and structural characteristics of definitions in ACL ARC. Our system performs at an F_1 of about 49% over the documents in the ACL ARC.

Subject Descriptors:
Natural Language Processing
Information Extraction
Language Resources
Digital libraries and archives

Keywords:
Definition Extraction, Conditional Random Fields, DefMiner, ACL Anthology Reference Corpus

Acknowledgement

I would like to thank my families and advisors. Without them, I would not have been able to complete this project.

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

Definitions are the basis for foundational knowledge in many fields. Through precise word, definitions clarify the scope and content of the terms that they define. They are an essential element in scientific discourse that merit the attention for automatic processing.

Scholars have long recognized the importance of definitions, compiling definitions into technical glossaries and dictionaries. The collective efforts of such lexicography studies have resulted in the modern dictionary and their contemporary online equivalents.

While large online dictionaries and encyclopedias exists, they are largely compiled manually, and as such, have the bottleneck of relying human editorship and effort. Even in successful crowdsourcing endeavors such as Wikipedia, where thousands of contributors collaborate together and create new content pages every day, the number of editors is far smaller than the number of active entities to be maintained. As evidence to this effect, a recent study (Frank & Soboroff, 2012) showed that on average, a news article in the most cited category (LivingPeople) typically gets cited in Wikipedia after more than 100 days.

More common to specialized of scholarship, glossary and definition resources are largely outdated or even unavailable.

A second limitation of such manual compiled resources is the lack of context. For example, the term "NLP" is almost always taken to mean "Natural Language Processing" in a paper on computational linguistics, however there are at least 13 different entries for it in Wikipedia. As a result, people consulting such resources have to expend additional effort to identify the

appropriate meaning of a term in context.

This motivates the focus of this thesis: to automatically extract definitions from scientific papers. A solution can help us tackle both limitations: first, most newly-minted technical terms are introduced in scientific publications, closing the bottleneck of staleness. Second, science is organized into disciplines. We can exploit this inherent structure to gather contextual information about a term and its definition. For example, the SIGIR conference is a conference for sharing research on information retrieval. Definitions extracted from SIGIR papers would be of relevance to explain information retrieval and search engine terminology.

The task of mining definitions has attracted a fair amount of research interest since 2000. The output of a definition extraction system can be directly used to generate glossaries or to answer definition questions.

The definition QA task has been popularized by the Text Retrieval Conference (TREC). The QA track started in 1999 and the performance of QA systems have increased significantly over the years. The objective of the TREC QA track is to build systems that can retrieve specific answers at runtime in response to posed questions. While the first few editions concentrated mostly on *factoid* questions – questions whose answers are just a single word or a short phrase – from TREC 2003 (Voorhees, 2003), definition questions were introduced. According to (Voorhees, 2003) in definition QA, an "exact answer" is not required and recall is more highly emphasized. So many systems use pattern-matching methods to retrieve a whole paragraph and then use other heuristics to reduce the amount of content to return.

One of the most direct applications for glossary generation systems is to build domain specific technical dictionaries. Since we can expect scientific papers to be a good source of newly-minted definitions, collecting terms and their definitions from these papers to form a glossary can provide us with helpful references. However, traditionally a large amount of time is required for domain experts to compile such glossaries. (Muresan & Klavans, 2002) asserted that automatic generation systems will significantly reduce the effort to build dictionaries.

Our work is closer to that of glossary generation. We seek to discover terms and their corresponding definitions from a large text corpus. Current work focuses on identifying sentences

which contain definitions. But to do this, we must first know what we are trying to identify. What exactly is a definition? This issue has been discussed the field of philosophy and has a number of interpretations. In this work, we define "definition" (Wikipedia, 2013) as:

> A passage that explains the meaning of a term (a word, phrase, or other set of symbols), or a type of thing. The term to be defined is the $definiendum$[1]. A $definiens$ is a cluster of words that defines that term[2].

Given the significant structure in definitions, we believe that it is important to go beyond merely identifying the entire definition and wish to isolate parts of the sentence that constitute the *definiendum* and *definiens*. We elect to study this problem in the domain of scholarly articles, as it is an evolving field where new terms and definitions are generated.

Lexicons we generate from such a system can be a platform on which to study the distribution and expression of definitions. It can also be used as an off-the-shelf training corpus for definition extraction task.

[1]We will use "definiendum" and "term" interchangeably when unambiguous.

[2]We often refer to the "definiens" simply as a "definition".

Chapter 2

Related Work

We now will discuss the previous methods adopted by the researchers to extract definitions from various sources. We summarize their methods into four main categories, namely rule-based approach (Muresan & Klavans, 2002), (Westerhout & Monachesi, 2007), machine learning approach (Fahmi & Bouma, 2006), (Westerhout, 2009), (Borg, Rosner, & Pace, 2009), hybrid approach (Blair-goldensohn, Mckeown, & Schlaikjer, 2003) and relation extraction approach (Navigli, Velardi, & Roma, 2010a), (Reiplinger, Schafer, & Wolska, 2012).

2.1 Rule-based Definition Extraction

Rule-based approach has the merit of being intuitive and efficient. The performance of the system is largely determined by the quality of the set of rules. This approach is adopted in the early research into definition extraction.

(Muresan & Klavans, 2002) developed a rule-based system to extract definitions from online medical articles. Their system consists of two major steps. The system first selects candidates using hand-crafted cue-phrases (e.g. *is defined as*, *is called*), then performs matching over the grammatical analysis to further filter unlikely definitional patterns from the set of sentences. Their grammatical analysis module identifies simple linguistic phenomena like relative clauses and appositions, which are common in definition sentences.

As part of the "European project Language Technology for eLearning" (LT4eL) Project,

(Westerhout & Monachesi, 2007) also developed a rule-based system to extract definitions from a Dutch eLearning corpus. In their system, they extensively exploit context phrases like "to be" or verbs indicating a definition. The best performing context they discovered is the verb *zijn* ('to be'), which achieves a F_2 score of 0.43. Besides context phrases, they also defined in total 303 grammar rules based on POS tags, chunk tags and marking of terms.

Although the manually crafted rules by linguistic experts can be very helpful in selecting accurate definition sentences, such systems typically suffer from low recall because definition sentences can often be expressed in a wide variety of ways. This makes it difficult to develop an exhaustive set of rules to locate all such sentences.

2.2 Machine Learning Approach

A typical way to mitigate the drawback of low recall for rule-based systems is to adopt a data-driven machine learning approach instead. This approach has been popularized since 2006 and is used by most following definition extraction systems.

(Fahmi & Bouma, 2006) use an external resource – a corpus consisting of Dutch versions of Wikipedia pages in the medical domain. Most first sentences in Wikipedia pages are a summary of the article and are also likely to be a definition sentence. They demonstrate a baseline approach which simply classifies every first sentence as a definition works well and gives an accuracy of 75.9%.

In order to improve the performance, they extract surface level features (bag-of-words, bi-grams, etc.), position of the sentence within the document, as well as syntactic features (type of determiner, position of the subject in the sentence) as evidence for model learning the classification process. Through their investigation of different learners – including naïve Bayes, maximum entropy and support vector machine (SVM) – they report best performance with the use of maximum entropy, achieving an accuracy of 92.2%.

A study of the statistical distribution of various grammatical items is included in their study. They find that *definiens* is more likely to be preceded by an indefinite article (71.8%) than a definite article (14.7%). However, we can imagine this feature will be noisy since many noun

phrases that are not *definiens* will also be preceded by an indefinite article. Their work does not conclusively show that the type of article alone helps in the classification process.

(Westerhout, 2009) also uses machine learning to augment a set of hand-written rules. A random forest classifier is used to exploit a set of linguistic and structural features. When selecting features, she is informed by (Fahmi & Bouma, 2006)'s study and includes the article and noun types into their feature set. Lexico-structural cues such as the layout of the text are also exploited. She evaluates the performance of different cue phrases that include the presence of "IS-A", other verbs, punctuations and pronouns. The highest F_2 score of 0.63 is reported for 'IS-A' pattern.

(Borg et al., 2009) propose a fully automated system to extract and rank definitions based on genetic algorithms and genetic programming. They define two sub-problems including 1) acquiring the relative importance of linguistic forms, and 2) learning of new linguistic forms. Starting with a set of 10 simple hand-coded features, such as having sequence "FW IS" (FW is a tag for foreign word) or containing keyword identified by the system, the system is able to learn simple rules such as "NN is a NN". However their system is optimized for similar "IS-A" patterns, as was used in (Westerhout, 2009). Their system, achieving an average F-measure of 0.25, also performs poorer than other machine learning systems which exploit more specific features.

2.3 Hybrid Approach

In (Blair-goldensohn et al., 2003), they introduce their system DefScriber, which uses a hybrid approach to tackle definition questions from the TREC Question-Answering (QA) track. After retrieving relevant documents for a given question, they used machine learning methods to identify "definitional predicates". Their system identifies three types of definitional predicates, namely genus, species and non-specific definitional (NSD) predicates. 383 out of 1,127 sentences in their example contain at least one definitional predicate. Finally they cluster the candidates using cosine distance measure and produce a list of cohesive and informative sentences addressing the definition question. In the TREC 2003 Definition QA evaluation their

system achieved an F score of 0.338; the median system performance among all competing entries was a much-lower 0.192.

The major contribution of their work is the predicate identification component, which involves learning partially specified syntax trees. This works well for genus-species sentences, where most of the definitions occur following a fixed set of regular expression patterns over part-of-speech tags, such as "DT? <TERM> is <GENUS> prep <SPECIES>". However, in real-life situation syntactic trees of definition sentences vary to large extent. We need a large set of definition sentences with various forms to train the model. If our corpus size is relatively small, this approach is not likely to work well.

2.4 Relation Extraction Approach

Beyond augmenting rule-based systems with machine learning approaches, more recent work such as that of (Navigli et al., 2010a) propose that a set of word-class lattices (WCL) be constructed for the classification model. Their approach consists of three steps. Firstly, each sentence is associated with a *star pattern*, where infrequent words in a sentence are replaced with a star '*'. For example, in the sentence:

In arts, a chiaroscuro is a monochrome picture.

is associated with the pattern:

a <TARGET> is a *.

Secondly, the sentences are clustered based on these star patterns. Lastly, a WCL is constructed for each cluster. A word lattice is a directed acyclic graph whose edges are labelled with a word and weight pair. The authors augmented this notation by allowing POS tags or the "TARGET" tag to replace the word. In their research, they manually annotated several thousand sentences from Wikipedia and marked the *definiendum*, *definiens* and *definitor* (the phrase or punctuation connecting the *definiendum* and the *definiens*) in the sentence. The best WCL classifier they construct achieves a high precision of over 99% and an F-measure of 0.75 on their self-constructed corpus of sentences drawn from Wikipedia.

In contrast to the majority supervised approaches, (Reiplinger et al., 2012) use an semi-supervised approach. They employ bootstrapping to extract glossary sentences from scientific articles in a corpus built from articles collected from the archives of the Association of Computational Linguistics (ACL)'s Anthology Reference Corpus (Bird et al., 2008). Their results show that bootstrapping can be useful for definition extraction. Staring with some initial term-definition pairs and hand-written patterns as seeds, their system can iteratively acquire new term-definition pairs and new patterns.

Table 2.1 presents an overview of prior work that we have reviewed, revealing choices with respect to their key aspects.

We note that all the systems reviewed only identify sentences containing definitions, and do not localize the definition to a specific part of the sentence, with the exception of (Muresan & Klavans, 2002). The results of their sentence classification task can be processed further to attempt to delimit the *defiendums* and *definiens*. but these works do not invest much effort in this final stage of processing, instead relying on simple hand-crafted rules. While (Navigli et al., 2010a) annotate their corpus to mark all *defiendums* and *definiens*, their evaluation restricts itself to definition sentence identification, to be comparable with other systems.

Compared to simply returning the definition sentence, being able to locate the exact *definiendum* and *definiens* in the sentence has several advantages. Firstly, we can generate structured "definition entities" instead of returning the unstructured sentence text. This can be more easily exploited by external (digital library) applications. Secondly, by isolating the *definiendum* and *definiens*, we can more closely study their specific linguistic characteristics.

We also believe that the presence of terms and definitions are helpful in the classification process of the sentences. So instead of adopting state-of-the-art system to find the definition sentence and then trying to recover the terms and definitions, we propose a system that directly extracts linguistic features from running text and assigns word-level annotation among "TERM", "DEFINITION" and "O" to each word. If we observe presence of a term and a definition in a sentence, we can be relatively certain that the sentence is a definition sentence.

Method	**Corpus**	**Granularity**	**Results** $\mathbf{P/R/F_1}$	**Author(s)**
rule-based, genetic algorithm	non-technical eLearning English texts	chunk unit	0.64/0.50/0.57	Muresan and Klavans (2002)
hybrid, ML + statistical analysis	AQUAINT Corpus of English News Text	sentence	f-measure 0.34	Blair-Goldensohn, McKeown and Schlaikjer (2003)
supervised learning (SVM)	Dutch Wikipedia.	sentence	sentence	Fahmi and Bouma (2006)
ML, rule-based on patterns	Multi-language corpus	sentence	f-measure of 0.79 for Is-a	Westerhout and Monachesi (2009)
evolutionary algorithm	eLearning English texts in field of ICT	sentence	1.00/0.51/0.68 for Is-a	Borg, Rosner and Pace (2009)
WCL	Wikipedia, ukWac	sentence	0.99/0.65/0.77 on Wikipedia corpus	Navigli and Velardi (2010)
Bootstrap	ACL ARC	NP pairs	acceptable agreement	Reiplinger et al. (2011)

Table 2.1: Overview of previous works on definition extraction

Chapter 3

Annotated Corpus Construction

Our central objective is to train a model for predicting the presence of terms (*definiendum*), definitions (*definiens*) and ordinary texts, which can be used to classify the text of an input scientific paper, assigning each word with a label from the 3-set of {"TERM", "DEFINITION" or "OUT"}. In order to train our classifier, we need a corpus of definition sentences where all the *definiendum* and *definiens* are marked out.

When we started out, we evaluated two sources of corpora which could be used for our work, namely 1) corpora for definition QA, and 2) manually-generated glossaries.

We looked at several definition QA corpora, including corpora from the previously reviewed TREC definition QA tasks (TREC, 2003). Unfortunately, we deemed these unsuitable, as the answers to definition questions are commonly just a short phrase instead of the original sentence, they do not annotate where the answers are derived from within the original source text.

We also considered manually-generated glossaries as sources for supervision. However, often list definitions which had been curated by human domain experts and their definitions are often worded differently from the original source text, and may even be derived from external text. For these reason, these are also not suitable for our purpose of learning models for definition occurrence.

Based on these negative findings in our initial literature search phrase, we thus decided to construct our own annotated corpus. Intuitively, online encyclopedias such as Wikipedia are good starting points for our corpus. They are large and often curated by human experts.

Therefore we constructed our pilot definition corpus "WikiDef" as a basis for all our initial experiments.

WikiDef. We extracted the sentences from the first paragraph of 30,000 Wikipedia pages. Wikipedia's style guide suggests that the first sentence of each article should contain the subject of the article (usually emboldened), and its accompanying definition. We randomly selected 1,000 of these sentences and manually annotated them to identify the terms being defined, as well as the definitions. We annotated at a speed of around 80 sentences/hr. It took us 20 hours to annotate and examine this corpus.

Since the objective of our project is to mine definition sentences from scientific articles, it is not sufficient for us just to develop our system against Wikipedia or general web resources. We therefore made use of the ACL Anthology Reference Corpus (ACL ARC), also leveraged by the related work reviewed previously. The ACL ARC consists of 10,912 articles up to February 2007 (Bird et al., 2008). To build a small development corpus for our work, we selected a small subset of 234 papers from this corpus – workshop papers published in 2000 – which we refer to as the W00 corpus.

The effort to manually annotate all 234 papers and locate the occurrences of definitions was significant. The set of papers we selected contains 38,243 sentences and 791,623 words.

To make the task feasible, we used several heuristic rules to create basic definition extractors. The idea is to make use of these extractors to filter out sentences which are not likely to contain definitions. We then proceed to annotate only the sentences that make it through this filtering step.

We built these basic, heuristic-based definition extractors informed by the previously reviewed research. Note that they are not meant to to extract definitions accurately but instead to identify a reduced set of candidate definitions for annotation. These extractors include:

1. a rule-based system similar to (Muresan & Klavans, 2002) , using a set of rules which we had handcrafted;

2. a system using keyphrases, based on our intuition that *definiendums* are likely to be keyphrases. A state-of-the-art key-phrase extraction system (KEA) (Witten, Paynter,

Frank, Gutwin, & Nevill-Manning, 1999) was used to extract a list of 20 keyphrases for each paper. The first two sentences which contain these keyphrases are returned as candidate sentences;

3. a simple sequence classifier using word and POS features trained on WikiDef corpus.

We took the union of the candidate sentences returned by these three extractors and obtained a pool of around 5,500 definition sentence candidates. Due to time constraint, we annotated half of the sentences to compile a final corpus of 2,512 sentences, 865 of which are real definition sentences and 1,647 are non-definition sentences. It took us around 70 hours to annotate this corpus.

Table 3.1 presents some statistics of the two corpora we have built.

Corpus Name	**No. Sentences**	**No. Definition Sentences**	**% Definition Sentences**	**Avg Sentence Length**	**Vocabulary Size**
WikiDef	1,024	293	28%	22	6,766
W00	2,356	802	34%	32	10,041

Table 3.1: Key statistics of the corpora constructed

By analysing the two corpora, we note that they have very different characteristics. The sentences in the W00 corpus tend to be more complex and varied in terms of sentence structure. Also, it has a larger vocabulary size than WikiDef. This reinforces our rationale as to why it was necessary for us to build and annotate the additional W00 corpus, rather than just relying on the WikiDef corpus, since our final goal is to be able to construct a classifier that can work on scientific papers.

Chapter 4

Implementation of Definition Extraction System

In this chapter we first describe conditional random fields method (CRFs), which is used as the classifier by our system. Then we introduce the features that we have used. Lastly, we will describe the preprocessing steps we applied to the corpus to isolate the language processing routines from the main task.

4.1 Sequence Classifier

We frame the problem of locating *definiendum* and *definiens* in a sentence as a sequence labelling problem at the word level. This contrasts with existing related work which largely attempts to classify entire sentences into definition and non-definition sentences.

The sequence labelling task has been intensively studied and has achieved satisfactory results in many NLP tasks. . To perform sequence labelling, researchers have applied several statistical models, including hidden Markov models (HMM), maximum-entropy Markov model (MEMM) and conditional random fields (CRFs). Although HMM and MEMM have been shown successful in tasks such as part-of-speech (POS) tagging (Ratnaparkhi & others, 1996) and named entity recognition (NER) (Zhou & Su, 2002), both methods suffer from the label-bias problem (Lafferty, McCallum, & Pereira, 2001). The label bias problem is due to per-state normalization

of transition scores. This means at each state, the outgoing edges has a probability sum of 1. For example in MEMM the label of the current word is solely determined by the sequence that precedes it:

$$P(S_1, ...S_n|O_1, ..., O_n) = \prod_{t=1}^{n} P(S_t|S_{t-1}, O_t))$$

Conditional Random Fields (CRFs) is a framework that builds probabilistic models for sequence labelling and segmentation tasks (Lafferty et al., 2001). It avoids the label-bias problem in a principled way. In contrast to MEMM which calculates the conditional probability at each state, CRFs avoids per-state normalization and uses a single joint probability of the entire label sequence given the observation sequence. This model can return an output sequence that maximizes the global joint probability for the whole observation sequence. CRFs have largely replaced the use of HMMs and MEMMs in most natural language tasks, and thus we also adopt them for use in our task.

In our execution of CRF learning, we incorporate evidence (features) drawn from several levels of granularity: from the word-, sentence- and document levels, not limiting ourselves to the window of previous N words. CRFs gives us this flexibility to encode such features which may not be independent[1].

4.2 Feature Extraction

In our experiments, we exploited a list of features ranging from basic lexical and orthography features, dictionary lookup features, corpus features (here, IDF), to more targeted shallow parse and dependency features. Table 4.1 is the full list of features we used in our experiments.

We briefly describe the rationale behind several of these feature classes. Lexical features such as the current and surrounding word or POS tags have been used widely in many sequence labelling tasks like POS tagging (Ratnaparkhi & others, 1996), named entity recognition (Zhou & Su, 2002). They provide fine-grained features which can be used to accurately assign labels to each word token.

[1]We use CRF++ (Kudo, 2005), an open source implementation of CRFs for segmenting/labelling sequential data, `http://code.google.com/p/crfpp/`.

Feature	**Description**
lexical	Including word, pos tag, stem word and suffix.
orthography	Whether the word is capitalized, mixed case, including period or hyphen.
dictionary	Whether the word is in the keyphrase dictionary.
idf	Discretized Inverse Document Frequency (IDF)
position	The section id, name and sentence relative position in the document.
first word	First word of the sentence.
has pronoun	Whether the sentence contains a pronoun
has acronym	Whether the sentence contains an acronym
surface pattern	Whether the sentence contain one of the hand-crafted pattern
shallow tag	The shallow parsing tag for each word.
shallow parsing pattern	If the shallow parsing sequence contains one or more patterns below: NP : NP NP is * NP NP is * NP that/of/which NP or NP known as NP NP (* NP) NP defined by/as * NP
ngram	If the shallow parsing sequence contains a frequent ngram collected from the WCL definition corpus
depend parrent	The dependency types where the current word is a child
depend child	The dependency types where the current word is a parent
root distance	distance of the current word to the root of the sentence
dependency path	The dependency path from the current word to the root of the sentence.
ancestor	The last two dependency type in the dependency path. (e.g. nn-dobj)

Table 4.1: Full list of features used in the experiments

We also observe that the terms being defined tend to possess regular orthography: they are more likely to be capitalized, surrounded by a pair of quotation marks, etc. We code features for such orthography to augment our feature set.

Dictionary lookup is a very helpful feature for the system to identify *definiendums*. Suppose we have a list of all the terms defined in a scientific article. We can then limit our focus to sentences just containing those terms. Unfortunately, it is not feasible to obtain such a list for large corpus of scientific publications. We therefore approximate it by using a list of keyphrases extracted by open keyphrase extraction system, KEA (Witten et al., 1999).

Most *definiendums* tend to appear relatively infrequent in the corpus, because few people will define commonly used words like "table". We believe we can improve the system's ability to capture *definiendums* by taking into account the frequency of the words. For simplicity, we use the inverse document frequency (IDF) instead of the more commonly used TF×IDF measure.

We include a set of position features in our classification system. These features include the section Id, section name, as well as sentences relative position in the whole document and its section. These features have been reported useful in keyphrase identification task (Nguyen & Kan, 2007). We also observe that the terms are more likely to be defined in the introduction or abstract section than conclusion section, and least likely in the reference section. Also, we find that definition sentences tend to appear in the beginning of their sections or subsections.

Although we try to avoid rely too much on the hand-crafted patterns, such patterns can provide us with important clues in definition identification. For example when we encounter the following sentence segment, we are relatively sure that the sentence contains a definition:

is defined as the

This is especially helpful when the training set is small and when we cannot cover all the definition formulations in our training set. So we maintain a list of accurate patterns in the form of surface word sequences and attempt to match them for each sentence. If the pattern is present in the sentence, the feature will be triggered.

The reason for us to extensively use the features based on shallow parsing is that features based on word form tend to be very sparse due to the large vocabulary size in English. On the

other hand, relying solely on POS tags alone does not give us information about the dependencies and relationships among the different words. When crafting the features, we firstly obtained the shallow parse sequence for 1,800 definition sentences drawn from the WCL corpus. We then clustered the sequences and summarized a list of patterns that are most productive.

Because the patterns we created cannot cover all the cases, we also created N-grams (N=4,5,6,7,8) of shallow parsing tags from the WCL definition sentence corpus. Intuitively if a sentence has a large overlapping with a known definition sentence in terms of shallow parsing sequence, it is likely to be definition sentence itself.

The classification of *definiens* is more difficult than that for *definiendums* because the structure of the *definiens* is markedly more varied. It is sometimes difficult for the sequence classifier to decide whether a specific word belongs to the *definiens*. We tackle this problem by extracting the dependency path from each word to the root of the sentence. This enables us to understand the relationship of the word to the rest of the sentence. However, the full dependency path can be very long, therefore making the feature space sparse. So we created two other features to represent the top-level dependency types. For example if the full dependency path is "conj_and-prep_of-nsubj-ccomp-root", then the ancestor of length one is "ccomp" and ancestor of length two is "nsubj-ccomp".

4.3 Corpus Preprocessing

In the course of conducting experiments, there is a need to repeatedly train and test on our data set. To reduce the execution time required for these iterations, we isolated the pre-processing involved from the main training and testing steps so that we do not have to repeat them unnecessarily.

After receiving an input article in plain text, the system first tokenizes the sentences in the article. For each sentence it extracts the section name and unique identifier (ID), as well as the relative position of the sentence within the document (the section feature is only applicable for the scientific document ACL ARC corpus). For each document, it generates a separate file which contains the original sentences (one sentence per line), POS tag sequences, shallow parse

tags (for each word), shallow parse sequences (for each sentence), named entity tags (for each word), dependency relations (one set of dependency per sentence) and metadata containing the section and sentence position information.

The linguistic features described above are obtained with the use of several popular NLP toolkits, including NLTK (Bird, 2006) for POS tagging, sentence and word tokenization; OpenNLP (Baldridge & Morton, 2004) for shallow parsing; and the Stanford Parser (catherine De Marneffe, Maccartney, & Manning, 2006) for dependency parsing.

Similar to the process of generating star patterns (Navigli et al., 2010a), during the shallow parsing stage, we maintain a list of signal words which we are interested in. The signal words are curated by the author manually and a sample list is listed in Appendix C. For these words, we return the original word in lower case instead of the shallow parsing tag. For example, the following sentence:

> A supercomputer is a computer at the front-line of current processing capacity, particularly speed of calculation.

Will result in the following tag sequence:

> a I-NP is a I-NP PP the I-NP of B-NP I-NP I-NP , ADVP B-NP of I-NP .

Chapter 5

Experiments and Discussions

The experimental results presented in this chapter are based on the W00 corpus (described in Chapter 3), unless specified otherwise. We chose to focus on the W00 corpus as it consists of a set of real research papers. This mirrors closely the target domain on which we hope to apply our work on.

5.1 Procedure

In this section we will give a brief overview of the preprocessing pipeline adopted by our system and the measure by which the output is evaluated.

We formatted the training and testing data and built classification models using CRF++. Ten-fold cross validation is performed to reduce variability of the data.

In both our corpora WikiDef and W00, the distribution of definitional and non-definitional sentences is skewed. Only 30% of the sentences within our corpora contain a definition. As this biases our classifier towards non-definitional sentences, for each iteration of the 10-fold cross-validation, we sampled from the original training data to remove approximately half of non-definitional sentences. This allows us to obtain a training corpus which contains a more balanced distribution of definitional and non-definitional sentences.

We evaluate our result on the word- as well as sentence levels. For each word token, our system predicts a tag, among "TERM", "DEF" (definition) or "O" (out). We calculate the

precision, recall, and F_1 scores for the different set of experiments and present our results in the next chapter. We also calculate the micro and macro-averaged F_1 scores for term and definition extraction F_{micro} and F_{macro}. F_{micro} assigns equal weight to every token while F_{macro} gives equal weight to every category. As the total number of definitions is much greater than that for the terms (roughly 6:1), we suggest the macro-averaged measure be used to normalize the contribution of term and definition identification results. For the sentence-level evaluation, we calculate the $P/R/F_1$ score based on whether the sentence is a definition sentence. We use sentence-level evaluation to have comparable results with the previous published work.

Aside from using different set of features, we also carry out experiments with different parameters for the machine learning module (CRF++). The two most influential parameters are the frequency threshold f and the degree of over-fitting c. By increasing the frequency threshold we can remove some features that influence few instances. The best parameter combination for our task is $f = 3$ and $c = 1.5$, which means we will only select the features that are triggered at least 3 times and we train a model that fits the training data slightly better than the default one ($c = 1.0$).

5.2 Experiments with Different Feature Sets.

We report experiment results using different feature sets in Table 5.1. The baseline system makes use of basic word and POS tag sequences as features, which are common to many baseline sequence labeling tasks.

We can see that most of the features results in an improvement to the baseline system, especially in terms of recall. Unexpectedly, the *position* features which include the section ID and the relative position of the sentence in the document, cause performance to drop.

The sentence ID feature we used is without normalization. The CRF++ module treats the sentence id feature simply as a string rather than numeric value. This makes the feature space very sparse. The system may well assert any sentence with ID 54 is a definition sentence when it observes two definition sentences with the same ID in the training data, which is likely to be due to pure chance. A better way to tackle it is to normalize the sentence ID feature and map

Features	**Term**			**Definition**			**Overall**	
	P	R	F_1	P	R	F_1	F_{micro}	F_{macro}
1: Baseline	0.49	0.34	0.40	0.41	0.49	0.45	0.45	0.44
2: 1+orthography	0.46	0.35	0.40	0.42	0.51	0.46	0.46	0.44
3: 2+dictionary	0.48	0.36	0.41	0.41	0.49	0.44	0.44	0.43
4: 3+idf	0.50	0.35	0.41	0.40	0.52	0.45	0.45	0.44
5: 4+position	0.47	0.37	0.42	0.36	0.48	0.41	0.41	0.41
6: 4+shallow parsing tag	**0.51**	0.38	0.43	0.41	0.50	0.45	0.46	0.46
7: 6+shallow parse pattern	0.50	0.40	**0.45**	0.42	0.52	0.47	0.47	0.47
8: 7+shallow parse ngrams	0.50	0.40	0.44	0.43	0.53	0.48	0.48	0.47
9: 8+surface pattern	0.49	0.39	0.44	0.43	0.53	0.48	0.48	0.47
10: 9+dependency	0.50	**0.41**	**0.45**	**0.45**	**0.54**	**0.49**	**0.49**	**0.48**

Table 5.1: Evaluation result on the W00 corpus with different feature sets.

it to a set of finite values, which we note as a limitation of our work that we hope to address in the future.

Our final system (Experiment 10 in Table 5.1) is able to boost the recall for term and definition classification by 7% and 5%, respectively, without sacrificing precision. The F_{macro} measure is improved from 0.44 to 0.48.

5.3 Experiment using 2-stage Classification

In a definition sentence, the $definiendum$ and the $definiens$ are usually connected by a copular verb or punctuation. They usually appear close to each other. So knowing the location of the *definiendum* is likely to help us to locate the *definiens* in the sentence. We therefore changed the setting of our experiment and split the classification into two stages. In the first stage, our system will identify all the *definiendums* ("terms") from the data set. We then incorporate this output into a second classifier which is used for *definiens* classification. We incorporate this knowledge by adding additional features. The new features used for the second stage of the

classification is shown in Table 5.2.

Feature	**Description**
follow term	Whether the current word is within window of 5 words from a term.
before term	If the current word appears before a term.
after term	If the current word appears after a term.
has term	If the current sentence contains a term.

Table 5.2: Additional features for 2-stage classification

Table 5.3 shows the experimental results of our experiments using the 2-stage classification pipeline. Surprisingly, there is a 10% increase in the precision of definition classification. With the 2-stage classifier, F_{macro} score further increases from 0.48 to 0.51. The results verify our intuition that term classification does help in definition classification. We believe this is because the new additional features helped to reduce the false positive classification rate. As definitions are longer and resemble ordinary English text, it is hard to tell them apart from ordinary sentences, leading a rise of the false positive rate. However, providing the classifier with information that no terms are present in a particular sentence, is a good hint that the rest of the sentence does not contain any definitions.

Features	**Term**			**Definition**			**Overall**	
	P	R	F_1	P	R	F_1	F_{micro}	F_{macro}
10. [From Table 5.1]	0.50	0.41	0.45	0.45	0.54	0.49	0.49	0.48
11. 10+2-stage	0.50	0.41	0.45	0.55	0.58	0.56	0.55	0.51
12. 10+oracle	0.50	0.41	0.45	0.79	0.82	0.80	*N/A*	*N/A*

Table 5.3: Evaluation result on the W00 corpus using 2-stage classification.

In order to study how much correct term classification results could inform definition identification, we did a follow-up experiment where we made use of oracular knowledge, considered as "cheating". Instead of using system output from the initial term classification stage, we generated the additional features for the second stage using the ground truth annotations from our corpus. Note that we do not evaluate the F_{micro} and F_{macro} score for the first stage in

this case. This experiment shows that term identification can have a strong positive influence on definition extraction, improving definition extraction from 49% to 80%, a leap of 31%. Not surprisingly, our current 2-stage classifier lags behind the oracular system by a large amount, largely due to the lackluster performance of our initial term classifier. It will be interesting to explore in future work how we can improve the performance of our term classifier so as to reap the benefits possible with our 2-stage classifier.

5.4 Tackling two Subtasks Separately

Logically we can divide our task into two subtasks: 1) locating the exact *definiendum* and *definiens* in a definitional sentence, and 2) to correctly identify definitional sentences. To study how well our system performs for the two tasks separately, we conducted two additional experiments.

For the first task, we first remove all the non-definition sentences from the corpus. Our purpose is to investigate the ability of the system to find the term and its definition in a sentence given the sentence is a definition sentence. We present the results in Table 5.4:

Term			**Definition**			**Overall**	
P	R	F_1	P	R	F_1	F_{micro}	F_{macro}
0.68	0.46	0.55	0.66	0.79	0.72	0.71	0.65

Table 5.4: Evaluation result on the W00 corpus after removing non-definition sentences.

Not surprisingly, this is an easier subtask compared with the initial compound task. To the best of my knowledge, no relevant published research has evaluated definition extraction task on this token level. So there is no immediately available result to compare our performance. We hope that our evaluation results can serve as a possible (simple) benchmark in this regard.

For the second task, we also want to investigate how well our system can identify definition sentences, which we remarked earlier is commonly found as the primary task in many prior works. The task to classify the sentence as definitional sentence or non-definitional sentence is also less ambitious compared to our system's compound task.

In order to compare our system with other definition extraction systems, we need to post-process the results to classify the sentence according to the output of our system. One method is to train two separate classifiers to recognize the *definiendum* and *definiens* separately, although the classification process may not necessarily be independent. Intuitively, if some part of the sentence is classified as a term while some disjoint part is classified as *definiens*, we are relatively confident that the sentence is a definition sentence. In Table 5.5 we present the evaluation result on simple models of this notion of two classifiers. The **union** classifier classifies the sentence to be definition sentence if the sentence contains either a term or definition (favoring recall); the **intersection** classifier selects the sentence only if it contains both (favoring precision).

Classifier	**P**	**R**	**F**$_1$
12. union classifier	0.60	0.84	0.70
13. intersection classifier	0.79	0.54	0.64

Table 5.5: Evaluation of different classifiers at the sentence level.

5.5 Experiment using the WCL Corpus

For the definitional sentence classification task, an F_1 score of 60% may not be able to convince the audience that our system is effective in extracting definitions. But we believe that the difficulty of the task is largely determined by the nature of the corpus. We believe that extracting definitions from scientific papers is much harder than from the leading sentences within Wikipedia articles. According to our knowledge, (Reiplinger et al., 2012) is the only attempt to extract such definitions from the ACL ARC corpus, which is a superset of our W00 corpus. It would be nice to have a directly comparable result with their work, but their evaluation method is mainly based on human judges and their reported coverage of 90% is only for a sample, short list of domain terms they defined in advance.

For the other related research reviewed earlier, unfortunately, we could neither obtain their source code nor the corpora used in their work, making comparative evaluation difficult. However, we experimented with our system on the whole WCL annotated corpus (Navigli, Velardi,

Ruiz-martnez, & Informtica, 2010b), reporting results in Table 5.6.

System	**Token Level**						**Sentence Level**		
	Term			**Definition**			**(Intersection)**		
	P	R	F_1	P	R	F_1	P	R	F_1
11. DefMiner	0.82	0.78	0.80	0.82	0.79	0.81	0.92	0.79	0.85
Navigli et al.	-	-	-	-	-	-	0.99	0.61	0.77

Table 5.6: Evaluation result on WCL Corpus

Compared to the result reported by (Navigli et al., 2010b), our system's F_1 improves over the competition by 8%. While the results show that our system's precision of 92% is lower than their 99%, recall is improved over their WCL-3 algorithm by almost 20%.

The experiment described above validate that our system can be used not only to extract $definiendums$ and $definiens$ from scientific publications, but also to classify definition sentences. The latter task is more well-studied and less aggressive than our target task.

Chapter 6

Application to Large Scientific Corpus

A key objective of our system is to be able to accurately extract definitions from scientific papers beyond our controlled dataset. In order to validate the usefulness of our system, we thus ran it on ACL ARC, which consists of 10,921 scholarly publications about computational linguistics (N.B. our W00 dataset is a subset of this larger corpus, so there is a small amount of overlap). We kept all the features in the best performing system reported in the last chapter except for the features based on dependency parsing. The reason is that generating dependency parses for the whole corpus requires significant amount of time. We trained a model using the whole of the W00 corpus and used the obtained, trained classifier to identify a list of definitions for each publication.

We then aggregate the definitions into volumes (e.g. J00 contains all the journal papers in year 2000). Instead of just returning the sentences, we generated valid XML files representing list of "definition objects". The XML files can be easily browsed by the user or exported by external applications. In each definition object, we define three properties: "sentence" which is the original sentence, "definiendum" the term being defined and "definiens" which is the part of sentence that defines the term. Because in some cases multiple terms are defined in a single sentence, we also give each *definiendum* and *definiens* a unique id. Below is an example of a

definition object.

```
<?xml version="1.0" encoding="UTF-8"?>
  <volume id="W01">
    ...
    <paper id="1315">
      <definition id="0">
        <sentence>Then the distance between A and B can be defined as : d ( A , B
        ) = ( |M A M AB | + |M B -M AB | ) / |M AB | In other words , the
        distance is the number of relation pairs that are not shared by the
        annotations normalized by the number that they R do share .</sentence>
        <definiendum id="0">distance</definiendum>
        <definiens id="0">d ( A , B ) = ( |M A M AB | + |M B -M AB | ) / |M AB |
        In other words</definiens>
        <definiens id="1">the number of relation pairs that are not shared by
        the annotations normalized by the number that they R do share</definiens>
      </definition>
    </paper>
    ....
  </volume>
```

In this sentence we can see that the term "distance" is defined in two ways, firstly by a formula, then by a text clause. In this example, DefMiner correctly extracts both *definiens* instances.

6.1 Manual Inspection of System Output

In order to quantitatively evaluate the definitions extracted by our system, we inspect the sentences and evaluate if the sentence is correctly annotated by the system. Due to the overhead in constructing manual relevance judgements, we limit our evaluation to workshop papers in 2001 and 2002. Table 6.1 presents the results. For each experiment, we report the total number of sentences extracted. "Total match" means each token in the sentence is labelled correctly. This is the most strict requirement. "Partial match" means the system can classify the sentence as definition sentence correctly using the intersection classifier, but that one or more tokens in the sentence is labelled incorrectly. The acceptance ratio can be understood as the precision

Corpus	**Total Documents**	**Extracted Sentences**	**Total Match**	**Partial Match**	**% Acceptance**
W01	153	695	405	87	70.8%
W02	273	1201	763	221	81.9%

Table 6.1: DefMiner evaluation on workshop papers from 2001 to 2002.

score of the sentence classifier.

From the result, we see that 70% of all the sentences extracted by our system are actual definition sentences. The majority of these definition sentences achieve a convincing "total match", meaning perfect correctness. We did not evaluate the recall for this task because we do not have the complete gold standard annotation for these two corpora – the relevance judgements were made only on the data returned by DefMiner as positive. We also see that DefMiner extracts, on average, four to five definitions per document. This number may seem small since each document contains around 200 sentences and we regard short phrase like "n is the number of clusters" also as a definition. However, in our task precision is preferred over recall because we would like to build a reasonably accurate list of definitions. While recall numbers are lower, the fact that we are processing a huge corpus allows us to mitigate this somewhat.

6.2 Analysing Extracted Definition Sentences

Although our system does not explicitly use surface patterns to filter sentences, it favors frequent copular patterns like "is a" or "is the". The reason is because they occur many times in the training set and thus are given more weight by the machine learning module.

In the following analysis, we take the experiment on the W01 corpus as an example. Among the 695 sentences extracted by the system, 147 of them contains derived patterns from "is a" or "is the". Among these, 125 sentences are "total match"es, and only 11 sentences are falsely identified as definitions. Thus, the "is a/is the" pattern achieves an acceptance ratio of 92.5%, far above the average figure.

As another example, the system also performs extremely well with recognizing "NP : S" as a definition sentence. 51 sentences are extracted using this signal and only 6 are false positive examples. By inspecting the sentences, we discover that most of them are well formatted definitions in the definition section or the title of the paper. It is quite common for a paper to have a title similar as the following:

> SHOE/TERM : A/DEF knowledge/DEF representation/DEF language/DEF for/DEF internet/DEF applications/DEF

When annotating the data, we found in most of the cases the term appears before the *definiens*. DefMiner, however, is sufficiently robust and is able to classify instances where the term appears after the *definiens*.

> This tree is input to the Logical Form module , which produces/DEF a/DEF deep/DEF syntactic/DEF representation/DEF of/DEF the/DEF input/DEF sentence/DEF , called the LF/TERM (Heidorn , G. E. , 2000) .

Our annotated corpus consisting of less than 800 definition sentences does not nearly cover all possible textual patterns that signal definition. An interesting angle of investigation is thus whether the system is able to identify the definition sentences which contain no pre-defined patterns. Our system is able to identify 71 definition sentences which do not contain any of the patterns encoded as a feature. Most of these sentences involves describing a system or procedure and uses verbs uncommon to definition sentences. For example our system extracted the following sentence:

> Estimation using the minimum/TERM description/TERM length/TERM principle/TERM involves/DEF finding/DEF a/DEF model/DEF which/DEF not/DEF only/DEF 'explains'/DEF the/DEF training/DEF material/DEF well/DEF ,/DEF but/DEF also/DEF is/DEF compact/DEF .

The clause "involves finding a model which not only 'explains' the training material well, but also is compact" does explain the meaning of the term "minimum description length principle".

But since no signal phrase is present, this sentence is likely to be missed by any rule-based definition extraction system or even human annotators.

We also look closely into the mistakes made by our system and summarized these into several categories. First, the system has the tendency to mark the first few tokens as "TERM" while the real term appears somewhere else in the sentence. For example in the following sentence:

> A PSS/TERM thus contains abstract linguistic values for "closed " features (tense/DEF ,/DEF mood/DEF ,/DEF voice/DEF ,/DEF number/DEF ,/DEF gender/DEF ,/DEF etc/DEF ./DEF

The term being defined is "closed features" instead of "PSS". "PSS" is likely to have received a high IDF and has capitalized orthography, and could therefore be misclassified as a term.

Second, due to the nature of sequence classification, the label of a token is influenced by the labels of the surrounding tokens. Thus the system will sometimes be confused when it encounters recursive definition or multiple definitions in a single sentence. In the following example:

> Similarly , 'I'/TERM refers to an/DEF interior/DEF character/DEF and/DEF 'L'/DEF indicates/DEF the/DEF last/DEF character/DEF of/DEF a/DEF word/DEF .

The sentence contains two parallel definitions. The classifier failed to classify 'L' as a separate term but rather took it as part of the *definiens*. Similarly, it is also not uncommon for the classifier to make mistakes with the boundary detection, especially for the *definiens* because they often span multiple clauses.

Another difficult problem faced by the classifier is lack of contextual information. In the following sentence:

> Again one could argue that the ability to convey such uncertainty and reliability information to a non-specialist/TERM is a/DEF key/DEF advantage/DEF of/DEF textual/DEF summaries/DEF over/DEF graphs/DEF .

If we just look at part of the sentence “a non-specialist is a key advantage of textual summaries over graphs”, without trying to understand the meaning of the sentence, we may well conclude that it is a definition sentence because of the cue phrase “is a”. But clearly the whole sentence is not a definition sentence.

Chapter 7

Gaining Insights from the Automatically Generated Lexicon

After we have obtained a lexicon from a large data set, we performed some analysis over this result. Here, we describe some interesting experiments we have carried out to gain insights from the compiled lexicon. Understanding more about the properties of definition sentences and their components will eventually help us to define better features to capture them in the longer term, as well as help further our knowledge of the natural language of definitions and the structure of scientific discourse. While the results are only indicative (partially due to the fact that we employed our DefMiner system to obtain the lexicon we are studying from, which undoubtably has errors), they are useful.

7.1 Statistical Studies into the Lexicon

The first question we want to study is where definitions typically appear in the document. This knowledge can help us to narrow down our focus when we search for definitions in an article. Our intuition is that authors usually define a term before he or she uses it, similar to anaphoric use of pronouns and other forms of natural language reference. Thus sentences with definitions should occur in earlier parts of articles. With the lexicon we obtained, it is easy to study the relationship between the occurrence of the definition and the position in the document.

Quantile	0	1	2	3	4	5	6	7	8	9
Count	5898	6258	6574	6202	5512	5020	4341	3593	2931	2233
%	12.1%	12.8%	13.5%	12.7%	11.3%	10.3%	8.9%	7.3%	6.0%	4.5%

Table 7.1: Distribution of extracted definitions.

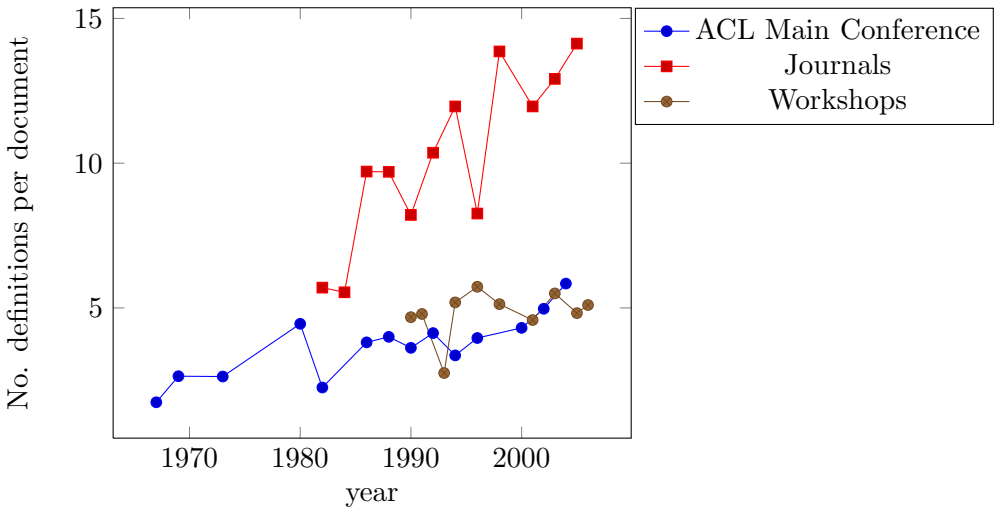

Figure 7.1: Definitions extracted per document for different types of articles

In order to formulate a standardized notion of location among the different documents with differing lengths, we simply divide each document into 10-quantiles containing roughly equal number of sentences. For example if a document has 100 sentences, the first 10 sentences will fall into the first quantile. For each definition sentence identified by the system, we note down the corresponding quantile where it comes from. The number of the extracted sentences located in each quantile is presented in Table 7.1. The results align closely with our intuition. The first three quantiles contribute almost 40% of all definitions while the last three contain only 17.8% of the definitions.

We next study if definitions appear as frequently in different types of scientific articles (e.g. journals, conference papers or workshop papers). We also want to investigate if there is a significant shift in the distribution of definitions across the years. In Figure 7.1, we present the number of definitions per document extracted by our system for these three different classes of publications.

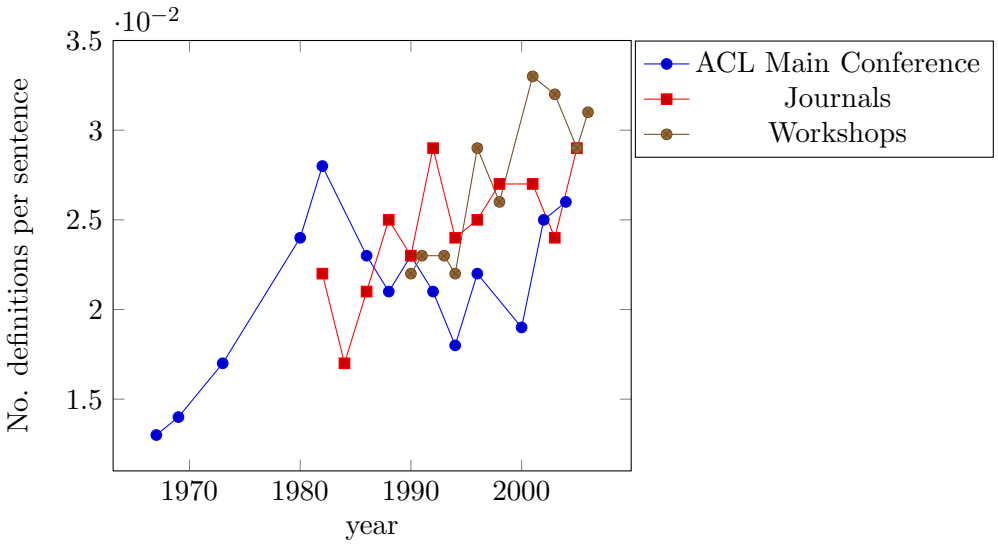

Figure 7.2: Definitions extracted per sentence for different types of articles

We note that a typical journal paper consistently contains more definitions than a conference paper or a workshop papers. However, from this view alone, we cannot conclude it is due to the difference in genre, as journal papers tend to be longer that the other two forms. To eliminate the influence of the length of documents, we normalize the number of extracted definitions in Figure 7.1 by the number of sentences in the document and re-plot the data as Figure 7.2.

In Figure 7.2, the three data series overlap each other, so we cannot conclude definitions appear more often in journal papers. However, as a side effect, we see that in all three publication forms, the number of definitions extracted per sentence has increased over time. The number of definitions presented in a ACL main conference paper, for instance, increased more than 100% in 40 years' time.

Specifically looking at the increasing number of definitions presented in journal papers, we want to study whether it is due to the sheer increase in the length of the journals. In Figure 7.3 we display the relationship between the number of sentence per document and percentage of sentences that are definition sentences.

We observe a large fluctuation in the graph, due to our limited analysed data, but note that when the length of the document is between 400 to 500 sentences, the percentage of definitions sentences tends to be larger. On the other hand, when the document is far longer (>550

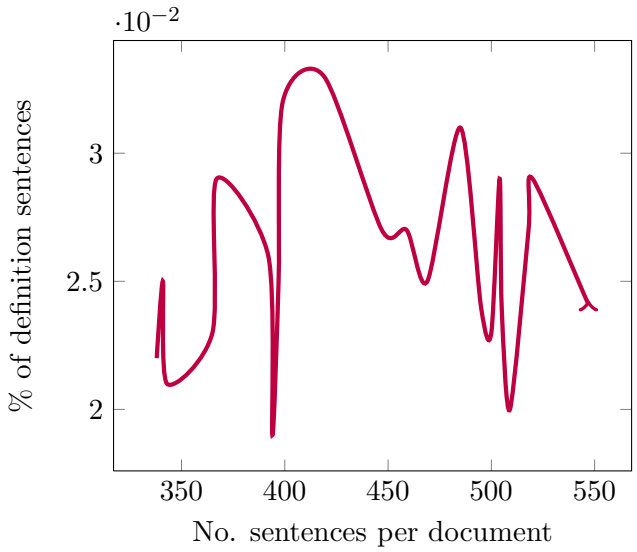

Figure 7.3: Correlation between document length and definition frequency for journals

sentences) or relatively shorter (<than 350 sentences), the proportion of definition sentences drops.

The increasing number of definitions in a document alone cannot show that more new knowledge is introduced. We also want to study how many of these definitions have not been introduced previously. In Figure 7.4, we plot for journal papers the total number of definitions of previously introduced terms in each year against the total number of new definitions. We say a definition is a *new definition* when the detected *definiendum* was not identified in any articles (not limited to journals) in previous years.

We see that the number of new terms being defined also increases with the years. But the increase is much slower than that for the total definitions. The red area denotes the definitions where the same term has been multiply-defined from the same or previous years as the current year under investigation.

7.2 Inspection of Frequently Defined Terms

Inspired by this finding, we want to look more closely at multiple definitions for a same term. We wish to investigate if they are referring to the same concept or are they talking about totally different things. Table 7.2 shows a sample list of highly frequently defined terms in the ACL

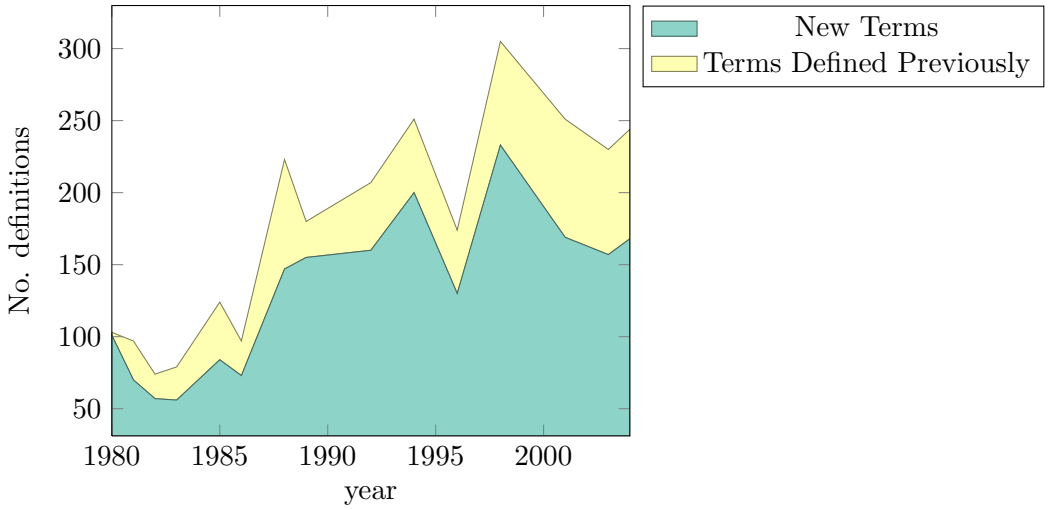

Figure 7.4: Newly introduced terms against terms defined in previous years

ARC.

We can easily categorize most of these terms into three categories. The first category is evaluation measures. As we have expected, the terms "precision" and "recall" appears at the top of the list. Many NLP researches use these popular measures as their evaluation method. Our system also identifies less frequently used measures like "accuracy" and "perplexity".

The second category names popular resources, corpora or datasets, including "WordNet", "FrameNet" and "PropBank".

The final category names popular algorithms like "HMM", "SVM", "LSA" and "CRF" and common NLP problems like "WSD". We assume that the definitions of such popular algorithms and problems will remain relatively unchanged. However, the definition for some other terms may vary in different context, for example the acronym "DM" may refer to totally different concepts. In Table 7.3 we present some definitions extracted for the terms "HMM","SVM" and "DM".

We can observe that although different authors have different focus, when they refer to terms like "HMM" and "SVM", the definitions are consistent to one another. However for a less well-known term "DM", the definitions tend to be diverse and vary more significantly. In the examples we showed in Table 7.3, the four definitions of "DM" refer to totally different concepts.

Term	**Count**
WordNet	292
precision / Precision	172
recall / Recall	167
NP / noun phrase	97
WSD/ Word Sense Disambiguation	60
SVM / SVMs	60
HMM / HMMs	54
LSA	57
POS	45
PCFG	43
FrameNet	38
DM	37
CRF /CRFs	29
idf / inverse document frequency	28
PropBank	27
CFG	25
accuracy	20
Perplexity	19

Table 7.2: Frequently defined terms

Term	Definition
HMM	The hidden Markov model (HMM) formulation is a powerful statistical framework that is well-suited to the speech recognition problem .
	HMM is a probabilistic finite state automaton used to model the probabilistic generation of sequential processes .
	HMM is one of the effective translation models (Vogel et al. , 1996) , which is easily scalable to very large training corpus .
DM	The detailed match (DM) is currently implemented as a beam-pruned depth-fast searched triphone tree .
	A discourse model (DM) is viewed as containing representations of entities , along with their properties and relations they participate in .
	Some type of words are very productive , such as numbers , DM (determinative measurement) , proper names .
	The DM is an extended and modified version of an earlier prototype developed by Jensen and Binot for the resolution of prepositional-phrase attachment ambiguities (Jensen & Bmot 1987) .
SVM	SVM is one of the binary classifiers based on maximum margin strategy introduced by Vapnik [16] .
	Similar to the PAUM , SVM is a maximal margin algorithm .
	The SVM (Vapnik , 1995) performs optimization to find a hyperplane with the largest margin that separates training examples into two classes .

Table 7.3: Extracted definitions for some terms

With this simple analysis we can have a glimpse of what the authors tend to define in scientific articles. Referring to the three categories we summarized above, the dataset and evaluation measures tend to be defined in the "experiment and discussion" section, while the techniques and the NLP problems can appear in "methodology" or "introduction" section. As an example, suppose that in a given paper, our system extracts the term "WSD" in the introduction, "CRF" in the methodology section, and "WordNet" in the experiment and discussion section, then we can guess that this paper is tackling WSD problem using CRF technique and the corpus is WordNet.

Chapter 8

Conclusion

We studied the task of mining definitions, going beyond just the identification of definition sentences. We propose to identify relevant parts of a sentence that contains a *definiendum* to be defined, and its *definiens*. Making use of sequence labelling with CRF, we show that we are able to achieve better results over previous published work as well as a suitable, word-only baseline. Unlike previous systems based on lexico-semantic patterns, our system is able to identify definition sentences when the patterns are absent. We demonstrate that with the aid of statistical, syntactical and linguistic features, our system is able to detect definitions without relying solely on traditional cue-words and heuristics. Though we have made use of a set of relatively simple features, our system result's are promising.

Observing the structure of our problem as a set of two intertwined subproblems of definiendum and definiens extraction, we also attempted a two-stage sequential learning architecture. Our experimental result confirms our hypothesis that the classification of definiendums help the system to more accurately find the definiens.

An additional contribution of our work are our two manually-annotated corpora for the definition extraction task, where each word token is annotated with a label of "TERM", "DEF" or "O". We plan to make these resources publicly available so that other researchers can benefit from our annotation efforts.

In Chapter 7, we present a short analysis made on definitions extracted by DefMiner from the *ACL Anthology Reference Corpus*. We observe that for the three types of papers (journal,

conference and workshop), the proportion of sentences that contain a definition increases. We also use the automatically acquired glossary to discover popularly defined terms, comparing their definitions across different documents, to find terms that are ambiguous as well as ones whose variant definitions represent the same core concept. Lastly, we summarize the definitions collectively, when we categorize definitions into the categories of "evaluation measures", "datasets" and "algorithms" and "common problems".

Our work enables several promising areas of future work. Much of previous work in this area has focused on definition sentence classification. If we can make use of their result to filter out the non-definition sentences, we will be able to further improve the performance of our system. In Table 5.4 we demonstrate that our system can identify the *definiendums* and *definiens* much more accurately on a set of definition sentences rather than a set of random sentences. Using the definitions collective extracted, we may also discover new definition patterns. If we have a sequence "... <definiendum> ... <definiens> ... ", then we can extract the context phrase which indicates the presence of a definition.

A second area of improvement is to incorporate use of third-party and public resources, to better "stand on the shoulders of giants". When I implemented the system, we were not aware of some existing resources. For example, I wrote my own code to extract the section name and ID, while existing state-of-the-art systems like ParsCit (Councill, Giles, & Kan, 2008) perform the same task more accurately. Leveraging such systems will better filter noise in the features, and enable downstream integration with existing digital library projects.

A final area is to pursue more in-depth analysis of the distributional and demographic properties of the automatically extracted lexica. We can use the lexicon obtained from different years to carry out trend prediction. Downstream systems may predict which term will become popular, or could alert an author if their definition of a term significantly differs from the original source.

References

Baldridge, J., & Morton, T. (2004). Opennlp.

Bird, S. (2006). Nltk: the natural language toolkit. *Proceedings of the COLING/ACL on Interactive presentation sessions* (pp. 69–72), Association for Computational Linguistics, 2006.

Bird, S., Dale, R., Dorr, B. J., Gibson, B., Joseph, M. T., yen Kan, M., Lee, D., Powley, B., Radev, D. R., & Tan, Y. F. (2008). The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics.

Blair-goldensohn, S., Mckeown, K. R., & Schlaikjer, A. H. (2003). A hybrid approach for qa track definitional questions. *In Proc. of the 12 th Annual Text Retrieval Conference* (pp. 185–192), 2003.

Borg, C., Rosner, M., & Pace, G. (2009). Evolutionary algorithms for definition extraction. *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09 (pp. 26–32), Stroudsburg, PA, USA, 2009: Association for Computational Linguistics.

catherine De Marneffe, M., Maccartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. *In Proc. Int?l Conf. on Language Resources and Evaluation (LREC)* (pp. 449–454), 2006.

Councill, I. G., Giles, C. L., & Kan, M.-Y. (2008). Parscit: An open-source crf reference string parsing package. *INTERNATIONAL LANGUAGE RESOURCES AND EVALUATION*, 2008: European Language Resources Association.

Fahmi, I., & Bouma, G. (2006). Learning to identify definitions using syntactic features. *Proceedings of the EACL workshop on Learning Structured Information in Natural Language Applications*, 2006.

Frank, J. R., & Soboroff, I. (2012). Knowledge based acceleration trec 2012.

Kudo, T. (2005). Crf++: Yet another crf toolkit. *Software available at http://crfpp. sourceforge. net*, , 2005.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01 (pp. 282–289), San Francisco, CA, USA, 2001: Morgan Kaufmann Publishers Inc.

Muresan, A., & Klavans, J. (2002). A method for automatically building and evaluating dictionary resources. *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2002.

Navigli, R., Velardi, P., & Roma, S. U. D. (2010a). Learning word-class lattices for definition and hypernym extraction.

Navigli, R., Velardi, P., Ruiz-martnez, J. M., & Informtica, F. D. (2010b). An annotated dataset for extracting definitions and hypernyms from the web.

Nguyen, T. D., & Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. *In Proceedings of International Conference on Asian Digital Libraries* (pp. 317–326), 2007: Springer.

Ratnaparkhi, A., et al. (1996). A maximum entropy model for part-of-speech tagging. *Proceedings of the conference on empirical methods in natural language processing*, Vol. 1 (pp. 133–142), Philadelphia, PA, 1996.

Reiplinger, M., Schafer, U., & Wolska, M. (2012). Extracting glossary sentences from scholarly articles: a comparative evaluation of pattern bootstrapping and deep analysis. *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, ACL '12 (pp. 55–65), Stroudsburg, PA, USA, 2012: Association for Computational Linguistics.

TREC (2003). Trec 2003 qa main task. Retrieved from `http://trec.nist.gov/data/qa/2003_qadata/main_task.html`. [Online; accessed 27-March-2013].

Voorhees, E. M. (2003). Overview of trec 2003. *Proceedings of TREC 2003* (pp. 1–13), 2003.

Westerhout, E. (2009). Definition extraction using linguistic and structural features. *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09 (pp. 61–67), Stroudsburg, PA, USA, 2009: Association for Computational Linguistics.

Westerhout, E., & Monachesi (2007). Extraction of dutch definitory contexts for elearning purposes. *Proceedings of CLIN*, January, 2007.

Wikipedia (2013). Definition — wikipedia, the free encyclopedia. Retrieved from `https://en.wikipedia.org/wiki/Definition`. [Online; accessed 27-March-2013].

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). Kea : Practical automatic keyphrase extraction. *Computer*, *pp*, 1999, 254–255.

Zhou, G., & Su, J. (2002). Named entity recognition using an hmm-based chunk tagger. *proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 473–480), Association for Computational Linguistics, 2002.

Appendix A

Hand-crafted Surface Patterns Used by the System

Below are the surface patterns encoded as features by our system.

<term>defined (as|by) <definition>

define(s)? <term>as <definition>

definition of <term><definition>

<term>a measure of <definition>

<term>is (a|the) <definition>(that|which|where|if)

<term>comprise(s)? <definition>

<term>consist(s)? of <definition>

<term>denote(s)? <definition>

<term>designate(s)? <definition>

<definition>(is|are|also) called <term>

<definition>(is|are|also) known as <term>

Appendix B

Keyphrases Identified by KEA

For the document *Extraction of Chinese Compound Words - An Experimental Study on a Very Large Corpus* (W00-1219), the KEA system extracts the following keyphrases:

Large Corpus

context dependency

mutual information

extracted compounds

lexicon

parameter settings

extraction of compounds

precision

Chinese Compound

information retrieval

Compound

query

MaxL

new compounds

retrieval

dependency

Appendix C

Signal Words Used during Shallow Parsing

a

the

which

that

define

describe

consist

call

refer

of

measure

equivalent

denote

designate

known