The Pennsylvania State University

The Graduate School

College of Engineering

AUTOMATICALLY EXTRACTING SEMANTIC SCHOLARLY ENTITIES FROM ACADEMIC PAPERS

A Thesis in

Computer Science and Engineering

by

Kaiyu Wang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2016

The thesis of Kaiyu Wang was reviewed and approved* by the following:

C. Lee Giles
Professor of Computer Science and Engineering
Thesis Advisor

Jesse L. Barlow
Professor of Computer Science and Engineering

Chitaranjan Das
Distinguished professor and interim head of Computer Science and Engineering

*Signatures are on file in the Graduate School

ABSTRACT

This thesis defines a new type of named entity– Semantic Scholarly Entity, which refers to the noun phrases in academic papers that deliver domain specific knowledge. The research goal is to automatically extract Semantic Scholarly Entities from computer and information science papers. We design a pipeline to extract candidate Semantic Scholarly Entities from computer science and information science academic papers and use supervised binary classification models to classify the candidate entities. 13 features are used and three types of machine learning models – Logistic Regression, Support Vector Machine (SVM) and Random Forest, are used and compared. The results demonstrate that SVM and Random Forest models outperform the Logistic Regression model. We also found that classic information extraction features like Term Frequency Inverse Document Frequency and Pointwise Mutual Information are the most important features in our classification models.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

Named entity recognition (NER) is an information extraction technique to locate and classify specific types of terms in a text (1). With the advancement of computer science innovation and web innovation, information is increasing procured from electronic records and sites through PCs, either online or offline. This change empowers individuals to utilize computer programs to organize, store or concentrate data from documents. However, the amount of online data is becoming enormous, and it is unthinkable for individuals to physically collect data they need and lead measurable investigation. In this way, individuals start to concentrate on effective and exact techniques to extract relevant data from electronic documents and sites.

In the 1990s, individuals saw the significance of recognizing important data units, such as individuals' names, organizations, dates and time. Being able to locate this data in documents brings significant advantages. To start with, web search tools scan a colossal volume of data online and return relevant sites and records to users on the basis of a couple of keywords. It is essential for a web search engine to precisely and proficiently locate specific information amid colossal volumes of online content. Second, individuals who conduct factual examinations may need to search for particular terms in a document corpus comprised of a great many records. For example, a patent office needs to check the seasons of references of celebrated makers in patent archives. In these circumstances, a name recognition algorithm can be utilized. NER issues play a more critical role in this type of data blasting.

There are now several mature programs for named element acknowledgment. General Architecture for Text Engineering (2) is a broadly utilized JAVA software of natural Language Processing tasks. It incorporates a tool called ANNIE, which is for information extraction

undertakings. This toolkit provides a series of helpful approaches for information extraction, including tokenisers, sentence splitters and part-of-speech taggers. ANNIE can be utilized, through a client graphical interface, to perceive named entities in unlimited content. Fig.1.1 shows the online demo of the ANNIE named-entity recognition system. The demo can identify names, organisations and locations in content. We can see that individual' names such as "Bill Gates" and "Paul Allen", association names such as "Microsoft" and locations like "Washington" are effectively perceived. It is also evident that blunders exist. One clear mistake is characterizing "Redmond" as a person's name, when in fact it is a location in the text.

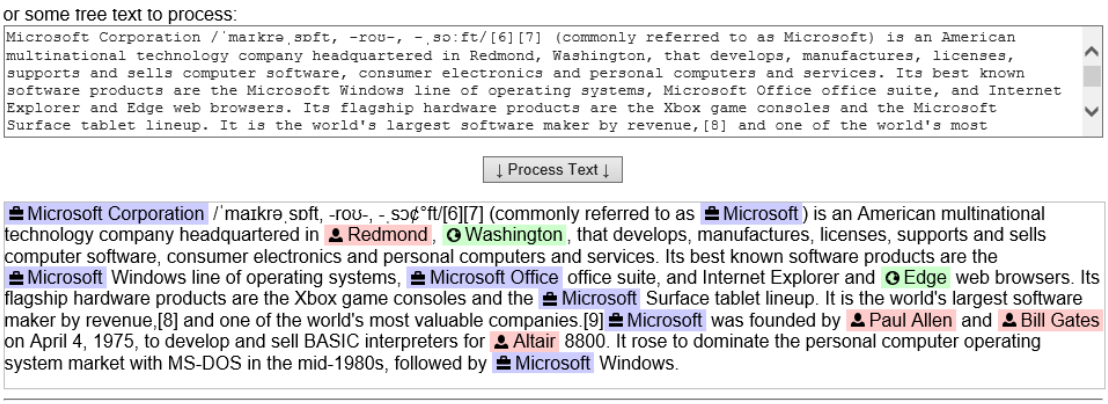

Fig.1.1 GATE named entity recognition system

Our research

Current research on NER problems focus on specific types of entities–individuals' names, organizations, locations and the entities that can be found on Wikipedia. However, in scientific papers, some terms are not present on Wikipedia, but they still convey scholarly domain knowledge. The paragraph below, which originates from the paper T*ensor Decomposition for Fast Parsing with Latent-Variable PCFGs* provides several illustrations:

> *There are various algorithms to perform CPD, such as alternating least squares, direct linear decomposition, alternating trilinear decomposition and pseudo alternating least squares. Most of these implementations treat the problem of identifying the approximate tensor as an optimization problem. These algorithms are not exact. Any of these implementations can be used in our approach. We note that the decomposition optimization problem is hard, and often has multiple local maxima. Therefore, the algorithms mentioned above are inexact.*

The terms "CPD", "direct linear decomposition", "alternating trilinear decomposition" and "optimization problem" are mathematical methodologies and concepts. They convey strong scholarly information, but they are not necessarily incorporated in Wikipedia and are not the names of persons, organisations or locations. These terms are quite interesting - they contain certain domain knowledge and provide us information about the contents of the text. They are known as Semantic Scholarly Entity (SSE) and a detailed definition and description will be outlined in section 3. The final goal of our project is to implement an intelligent semantic digital library search engine that can locate scholarly entities among a large scientific paper corpus. Our motivation is that people's interests are not limited to traditional named entities, and sometimes

people need to find scientific papers, or concepts in a scientific paper that relate to the entities that convey strong scholarly information and domain knowledge. As far as we know, no previous studies have examined this issue yet.

The foundation of a semantic scholarly entity search engine is successfully recognizing semantic scholarly entities in scientific papers. This is the immediate goal of this research, as well as the topic of this thesis. This study adopts a supervised machine learning approach. Based on a large volume of computer science research papers, we extract a candidate semantic scholarly entity list for each paper in light of part-of-speech rules. Then we calculate word-level, document-level and corpus-level features for each candidate scholarly entity. Next, a certain amount of candidate entities are labeled based on our domain knowledge to construct the training dataset. Finally, supervised machine learning algorithms are used to build our semantic scholarly entity extraction models. This is a typical binary classification model –when presented with an unseen candidate scholarly entity, the model can automatically determine whether it is a real scholarly entity or not.

One problem with this semantic scholarly entity extraction method is that we need dataset, necessitating a tedious and time-consuming manual labeling process. Some recent research works on NER problems attempted to use the unsupervised approach, such as (3) and (4). However, their focus was on traditional named entities (facts such as names of famous people). Our project defines a new concept - Semantic Scholarly Entity, which currently has no existing extraction rules, templates or databases. Each term will need to be manually examined to determine whether or not it is a Semantic Scholarly Entity. Therefore, we think it is difficult and meaningless to use unsupervised learning to address such a new problem.

In this thesis, we make following contributions:

- Define a new type of entity – Semantic Scholarly Entity, which is different from the named-entities in other research works.

- Propose an approach to extract candidate Semantic Scholarly Entity from computer science research papers. Create a pipeline and a set of rules.
- Propose 13 features and build supervised classification models to recognize Semantic Scholarly Entities in Computer Science papers. Evaluate each model to determine the optimal machine learning algorithm and sampling strategy.
- Create software capable of extracting candidate Semantic Scholarly Entity and calculate word-level, document-level and corpus-level features with a multi-process strategy based on a large file corpus.

Thesis Structure

The remainder of this thesis is organized as follows. In chapter 2, we review some relevant research on keyword extraction and named-entity recognition problems. In chapter 3, we introduce preliminary topics, including the definition of semantic scholarly entity and the tools used in the study. Chapter 4 provides a detailed outline of our candidate scholarly entities extraction approach and the features used. The labeling of the training dataset is described in chapter 5.Chapter 6 provides a description of the sampling strategies used. In chapter 7, we describe the process of building machine learning models, evaluate the models and report our findings. Chapter 8 will provide conclusions and make recommendations for future research.

Chapter 2 Literature Review

This chapter reviews previous literature related to keyword and named-entity extraction. Note that our project does not aim to extract any type of keywords. But we may use some of the techniques in keywords extraction. Thus, we provide reviews of keyword extraction works with an emphasis on the effective and widely-used features.

Keyword extraction techniques

According to Wikipedia, key word extraction is the process of automatically identifying terms that best describe the topic of the document (5). People sometimes want to know the content of a textual document. However, reading and summarizing the key points of a large volume of documents is time consuming and sometimes even impossible for humans. With the development of computer-based text analysis techniques, researchers are focusing on methods that can automatically identify the focus and main themes of a document, leading to keyword extraction techniques.

Determining whether a set of words accurately represent the topics of a document is challenging because natural language is complicated, as stated by Brian Lott (6). Nevertheless, many word importance measures have been proposed and used.

Early research focused on applying term frequency to a document, to select the words that reflect the overall contents of the text. For instance, some of the most frequent noun terms in a document may convey some message about the textual content. Term frequency (TF) is such a measure. It simply counts the occurrence of a term in a text. People can then search for terms in a document based on their TF measure and select a number of top terms to interpret the document.

In the 1970s, the idea of statistically analyzing the frequency of terms within a document, as well as the frequency of these terms in the whole corpus, appeared. A typical example of this idea is the Term Frequency Inverse Document Frequency (TFIDF) approach that was developed by K.S Jones (7). The basic idea of TFIDF is that a term may be important to a document if it occurs frequently, while the importance of the term may also be weakened by increased occurrence in the whole corpus. TFIDF usually takes the form of a measure of the frequency of occurrence of a term within one document, along with a measure of the frequency of the term's occurrence in the whole corpus. TFIDF facilitates the elimination of frequently used by trivial terms such as common nouns (people, conference, we, etc). According to K.S.Jones, TFIDF is still a frequently used feature today due to its effectiveness and simplicity.

Some measures use word co-occurrence properties. Y. Matsuo and M. Ishizuka propose a keyword extraction algorithm that uses the co-occurrence distribution between the frequently occurring terms and other terms. According to Matsuo, "if the distribution of co-occurrence between a particular term and the frequent terms is biased to a subset of the frequent terms, this particular term is likely to be an important term"(8). A benefit of using this algorithm is that it only needs to examine a single document, with no need to consider a large corpus. Yukio Ohsawa,NelsE.Benson and Masahiko Yachida introduced a graph-based, key-word extraction algorithm called "KeyGraph" (9). Their algorithm first creates a graph, in which each node corresponds with a term in the document and each edge represents a co-occurrence relation between two terms, to represent the document. Then they recognize maximal singly-connected subgraphs in the document graph (*clusters*). The terms that connect these clusters are extracted as keywords. This algorithm makes use of properties of complex graphs. Similar to Y. Matsuo's approach, this approach does not need to examine the document corpus.

Lexical chain is a widely-used technique in text summarization. However, Gonenc Ercan and Ilyas Cicekli proposed a lexical chain based keyword extraction algorithm (10). A lexical

chain for a text contains a subset of the words in a text and these words are semantically related. Gonenc and Ilyas apply scores to the terms derived from lexical chains of text and combine the scores with plain-text based features to generate feature vectors. A decision-tree learning algorithm is used to build machine learning models, which is chosen by experiments and comparison among different supervised learning algorithms. They achieved an accuracy rate of 64% and claim to be the first to use lexical-chain algorithms for keyword extraction purposes.

Our project does not aim to extract keywords of scientific papers. We aim to extract scholarly semantic entities. Scholarly semantic entities may or may not be the keywords of a scientific paper. However, the methods used in keyword extraction are helpful for scholarly entity extraction.

Named Entity Recognition and Classification

According to David Nadeau and Satoshi Sekine, people recognized by the mid-1990s that it is vital to recognising information such as names and numeric expressions, including people, organizations, locations, dates and money. Identifying the references to these entities is regarded as one of the most important tasks in Information Retrieval. This kind of work is known as Named Entity Recognition and Classification (NERC) (11).

Named entities refer to only those entities that accord with the rule of rigid designators. Rigid designators was defined by S. Kripke (1982), and refers to terms that designate the same thing in all possible worlds in which that thing exists and does not designate anything else in those possible worlds in which that thing does not exist (12). David and Satoshi claim that although the targeting terms in NERC problems are temporal and some numeric expressions, some of these terms are not valid rigid designators. In some cases, the NE definition is loosened

for practical reasons. Most studied term types are three types of “proper names”: names of “persons”, “locations” and “organizations”, which are known as “enamex”. Each category could have subtypes. For instance, “location” can be divided into subtypes such as city, county and country. “Person” can be divided into “politician” and “entertainer” (11).

Certain proper names are outside the classic“enamex”. The type “miscellaneous” is a word used in CONLL conferences, and includes subtypes like “timex” (dates and time) and numex (money and percent). Research interest may also influence the targeting types of terms. For example, interest in Bioinformatics led to many works dedicated to types such as “protein”, “DNA” and “RNA”. Certain studies do not limit the range of terms under extraction. David and Satoshi, S. Sekine and Nobata (2004) tried to cover most of the frequent name types appearing in newspapers (11).

An essential part of the NERC problem is to recognized unknown named entities in articles, a process known as “learning”. In old works, handcrafted rules are used for learning, while most recent works use machine learning (ML) methods. When using ML techniques to recognized named entities, each sample is associated with a feature vector, from which Machine Learning algorithms “learn” to generate classification rules. “Features are characteristic attributes of words designed for algorithmic consumption”, explained David and Satoshi in (11). We categorize the learning strategies of recent NERC works into two types: parsing paper text to acquire features and mapping extracted words onto some knowledgebase to acquire features. In the following two sections, the features used will be analyzed.

Named Entity Recognition based on text data

When people acquire the features based on pure text, they only need a paper file corpus. Based on David and Satoshi's argument in (11), we classify the features acquired through this strategy into three categories– word-level features, list-lookup level features and corpus-level features.

Word-level features relate to the composition of the characters of candidate entities, such as the case of each letter in the word, the existence of digits and the prefix or suffix of the word. One example is a Boolean value to represent whether the ending of a word is "ist", as human professions usually end with an "ist". Another widely-used example is to consider the case of the letters in a term. A term with an upper-cased first letter could be a name. Similarly, a term where all letters are in upper-case may be a concept abbreviation.

List-lookup features are usually introduced by the inclusion of a candidate entity in a word list. List inclusion is way to express the relation "is a". For instance, people can generate a list of words that appear frequently in organisation names, and check whether a candidate entity contains the words in the list to decide the probability of the word being a company name. David D McDonald used the term "associates" to identify organisations' names (13). However, sometimes terms with similar meanings may show spelling variations. Strategies can be used to capture words with similar meanings. Popular strategies include stemming and edit-distance.

Document and corpus-level features are related to the content and structure of a document or documents in a corpus. Some studies try to recognized multiple occurrences of candidate entities in the document and derive features from them. Certain works even used the statistics of the candidate entities in the entire corpus to make features. For instance, Christine Thielen identifies different variations of the same word that appears in one document for German

proper name recognition. Term frequency, document frequency and term-frequency inverse document frequency, which we mentioned in section 2.1, may also be useful features.

Named Entity Recognition based on external knowledgebase

Knowledgebase is a technology that was originally used to sure complex information for computer system usage. People could look up the meanings of terms with the help of knowledgebase. Several knowledgebase services have appeared recently, of which Wikipedia is a good example. Some research works on NERC problems try to create knowledgebase-related features. One good example is Jun' ichi Kazama and KentaroTorisawa's work in 2007 (14). They exploit Wikipedia to recognized named entities in their work. They extract word sequences from articles and find corresponding entities in Wikipedia. After converting a candidate word sequence in the article into a Wikipedia entity name, they try to find the Wikipedia category label of word sequence and use it as a feature of the NE tagger of the candidate word. They use a Conditional Random Field algorithm and compare the F-measure of the model using Wikipedia-related features and traditional NE features. The research found that Wikipedia labels can boostNER accuracy.

Machine Learning Techniques in NER Problems

Three types of ML techniques were studied in NERC – supervised learning, semi-supervised learning and unsupervised learning.

According to David and Satoshi, the majority of recent research uses supervised machine learning techniques in NERC problems (11). The theory behind supervised ML technique is to "train" learning models with a amount of labeled samples and to use this trained model to

automatically label unknown terms. One issue with supervised learning is the large amount of data labeling that is necessary, which means that researchers must know exactly which entities are named entities. When a large training corpus is not available, semi-supervised learning and unsupervised learning may be used. The most popular semi-supervised learning technique is *bootstrapping*, which requires some degree of supervision. In NERC problems, semi-supervised learning can be applied as per the following example. First a small amount of specific entities are used as input "examples" to the learning system. The system then searches the corpus to locate each entity and analyze the features of the surrounding context of the entity. Using these learned features, the system can extract similar entities automatically. As only a small amount of training data is necessary, semi-supervised learning methods overcome the disadvantage of supervised learning algorithms. According to David and Satoshi's paper, the semi-supervised algorithm outperforms rival supervised approaches (11).Clustering is the most popular unsupervised learning approach. For instance, it is possible to cluster sentence groups with similar context and extract named entities from these clustered sentences. Some other unsupervised learning techniques may require pre-computed or external resources and statistics such as word type library or Pointwise Mutual Information.

Chapter 3

Preliminaries

In this chapter, a definition of Semantic Scholarly Entity (SSE) will be provided, along with the problem definition.

Semantic Scholarly Entity

This research is interested in computer science research paper terms that could convey scholarly information and domain knowledge. A semantic scholarly entity consists of one or more individual words and has the following characteristics:

(1) A semantic scholarly entity must be a word or phrase in a text that refers to a meaningful thing.

(2) A semantic scholarly entity must be a term that describes:

- A kind of tool
- A method or an approach, including algorithms and theorems
- A concept or an idea
- A dataset
- A process
- A parameter

This study uses Python toolkits to parse candidate noun n grams in academic papers. The parser allow me to create “grammars” to guarantee that the parsed phrase obeys certain rules. Then we manually decide whether a candidate entity is a semantic scholarly entity based on our knowledge. Chapter 5 provides a detailed description of how to label candidate scholarly entities.

Tools

This chapter describes the software tools and techniques used in this project.

Natural Language Toolkits:

Natural Language Toolkits (NLTK) is a widely-used Python package for human-language text processing (15). It provides functions for part-of-speech tagging, gram parser, sentence tokenization, etc. This project used the following three NLTL functions:

1 **nltk.word_tokenize()**: This function splits strings into individual words and punctuations. It won't split compound words connected with '-'or '_'.

2 **nltk.pos_tag()**: This function analyses the structure of a sequence of words, classifies the words into their parts of speech and labels them, in a process referred to as parts-of-speech tagging. The function results in a list of tuples, in which each word has a part-of-speech tag. Below is an example of the returned list of the sentence "France is a developed European country with more than 60 million people."

[(u'France', 'NNP'), (u'is', 'VBZ'), (u'a', 'DT'), (u'developed', 'JJ'), (u'European', 'JJ'), (u'country', 'NN'), (u'with', 'IN'), (u'more', 'JJR'), (u'than', 'IN'), (u'60', 'CD'), (u'million', 'CD'), (u'people', 'NNS'), (u'.', '.')]

In each tuple of the list, the second element is the tagger, which indicates the part-of-speech of the word. NLTK uses the Penn Treebank part-of-speech tags shown in the appendix.

3 **nltk.RegexpParser.parse()**: This is the grammar-based chunk parser provided by NLTK. Users can create a set of regular expressions to specify which words should be parsed. For

example, the following regular expression parse three grams that have the structure adjective + noun or adjective + noun:

{ < JJ><NN | JJ ><NN> }

This function also uses Penn Treebank part-of-speech tags. Thus, JJ represents adjectives, while NN represents nouns. "|" represent *or,* indicating that the second word can be either a noun or an adjective. This function may be used on a list of tagged words, which may be the result of nltk.pos_tag function, and it returns a NLTK tree structure. In NLTK trees, each node is labeled and has several children nodes, which are recursively tree structures. The parsed grams are stored in these children nodes. The following is a returned visualized NLTK tree of the input sentence "*France is a developed European country with more than 60 million people.*" with the grammar *2gram: {<JJ><NN | NNP>}*:

```
(S
 France/NNP
 is/VBZ
 a/DT
 developed/JJ
(2 gram European/JJ country/NN)
 with/IN
 more/JJR
 than/IN
60/CD
million/CD
people/NNS
./.)
```

In the visualised tree structure above, the parsed candidate three gram is "European country", which is labeled by a created name "1 gram". Note that we need to traverse the tree to gather the parsed grams according to their labels.

Stanford Part-Of-Speech Tagger

According to Stanford Natural Language Processing Group (16), Stanford Part-Of-Speech Tagger is a Java implementation of the log-linear part-of-speech taggers, which apply the algorithms created byKristina Toutanova and ChristopherD. Manning (17).The software is very similar to NLTK POSTagger, where the string must first be tokenised and a list of words used as the input. The returned result is a list of tuples of words and their part-of-speech tag. For example, the following is the result list of the sentence "*France is a developed European country with more than 60 million people"*:

[(u'France', u'NNP'), (u'is', u'VBZ'), (u'a', u'DT'), (u'developed', u'JJ'), (u'European', u'JJ'), (u'country', u'NN'), (u'with', u'IN'), (u'more', u'JJR'), (u'than', u'IN'), (u'60', u'CD'), (u'million', u'CD'), (u'people', u'NNS'), (u'.', u'.')]

Note that the resulting format is the same as NLTK postagger, enabling the application of the NLTK parser on Stanford postagger.

Grobid

"GROBID is a machine learning library for extracting, parsing and re-structuring raw documents such as PDF into structured TEI-encoded documents with a focus on Scientific and technical publications" (18). This Java package can recognize the structure of a scientific paper and separate it into different sections. Grobid returns an XML file encoded in Text Encoding Initiative (TEI), which is a standard for the representation of texts in digital form (19). In other words, Grobid returns unified – tagged XML files for any PDF scientific papers. This enables the extraction of specific sections of text from scientific papers. This project is interested in the

position of a candidate entity. By using Grobid, we can accurately identify whether a candidate entity occurs in the title, abstract, body and the reference of a paper.

Scikit-learn

Scikit-Learn is an open source, simple and efficient Python toolkits for data analysis and data mining (20). It implements several commonly used data-mining and machine learning related algorithms. The algorithms range from regression, classification, clustering, dimension reduction and even data preprocessing.

As this study does not aim to create any new machine-learning algorithms, the machine learning implementations provided in Scikit-Learn are used to build prediction models. Scikit-Learn provides implementation of almost all types of popular classification algorithms, including Support Vector Machine, Decision Trees, Random Forests, Nearest Neighbors. It also implements some machine learning model building strategies, such as cross-validation and feature selection. The machine-learning algorithms are implemented as Python functions, whereby data and parameters are passed as function variables. Scikit-Learn allows us to set the parameters of some machine learning models. For instance, we can set the penalty C and the kernel we use in Support Vector Machine by modifying the parameter *C* and *kernel* when calling the function. This research used the machine-learning algorithm in addition to the cross-validation implementations provided by Scikit-Learn.

Chapter 4
Entity Extraction approach and Features

Our study aims to build a semantic scholarly entity extraction model for computer science field. Flow chart Figure 3.1 displays the project pipeline.

This section introduces the first three steps in the flowchart above. In a binary classification model, the most crucial aspect is to choose samples and create a feature vector. We will introduce our candidate scholarly entity extraction methods and then describe the features created for each candidate entity.

PDF articles to XML file conversion

As outlined previously, Grobid is used to convert PDF files into XML files that include all of the textual elements of the original articles. XML is a markup language that defines a set of rules for the format of each file. Grobid could divide a scientific paper into different sections, with each section tagged with its section title. This enables the extraction of sections as necessary. Usually, a scientific paper may include the following sections or elements:

Title: The title of the paper

Authors: Author information of the paper, including their names and emails

Abstract: The abstract of the paper

Body: The main body of the paper. This may include introduction, related works, approach, evaluation and so on

Reference: The information of the referenced papers

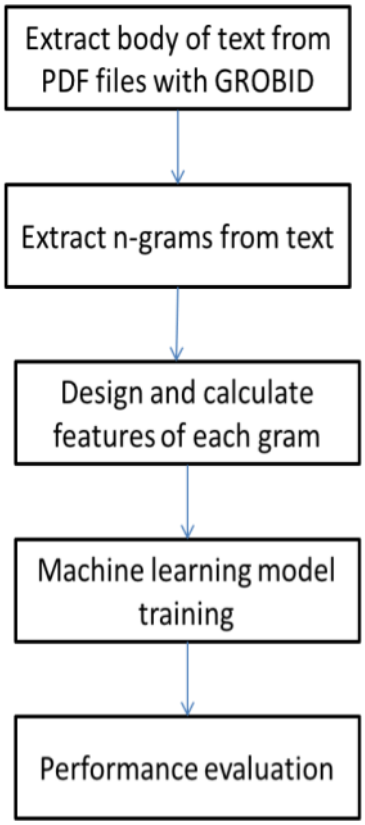

Figure 4.1 Project pipeline

This project focuses on the text in the title, the abstract and the body. Text from the reference is not considered, because there are only paper titles and names in reference, and these texts are not created by the authors of the paper.

Candidate semantic scholarly entity extraction

Flowchart figure 3.2 shows the pipeline of our candidate entity extraction approach. In our project, we extract n grams that conform to some rules from article texts and use them as candidate scholarly entities. We will introduce n gram first, and then tokenisation and part-of-speech tagging will be discussed.

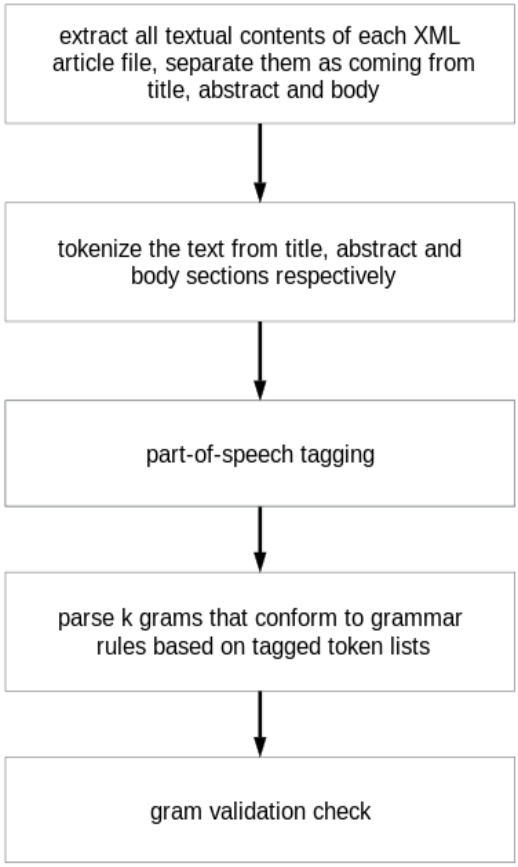

Figure 4.2 Pipeline of candidate semantic scholarly entity extraction

n grams

In text mining, a n gram refers to a contiguous sequence of *n* words from a given sequence of text or speech. Simply speaking, *grams* are phrases in the text. In this research, N-gram represents a phrase that consists of k words. For example, take the sentence below:

> *Recent Web utilization data for Picture Quest indicate that of the 10% of users from outside the United States, a significant portion come from Spanish-speaking, French-speaking, and German-speaking countries.* (Cross-Language Multimedia Information Retrieval, Sharon Flank)

In the example, “Recent Web utilization data” is a four-gram, “German-speaking countries” is a two gram. As explained in chapter 1, scholarly entities are phrases. Therefore, we extract n grams from academic papers as scholarly entity candidates. In this project, the maximum value of K is 5.

However, we argue that not every n gram could be a candidate scholarly entity. First, a scholar entity must be a noun, meaning that the last word in the candidate n gram should be a noun, and the gram should not contain verbs. Second, there shouldn’t be any numbers or sentence-splicing punctuations such as commas and periods. For instance, *a significant portion come from* is a 5 gram, but does not qualify as a scholarly entity candidate as it is not a noun. Another example is a three-gram *10% of users*, which contains a number and is not a scholarly entity at all. We can generate a set of “grammar” for the extracted grams to regulate the structure of the phrases:

Unigram: Noun

Two-gram: Adjective or noun + noun

Three-gram: Adjective or noun + adjective or noun + noun

Four-gram: Adjective or noun + adjective or noun + adjective or noun + noun

Five-gram: Adjective or noun + adjective or noun + adjective or noun + adjective or noun + noun

Natural Language Toolkit (NLTK) is a famous Python-based human language processing toolkit. It facilitates the parsing of grams from textual files based on specific grammars. This project compares NLTK Part-Of-Speech tagger as well as Stanford Log-linear Part-Of-Speech Tagger to parse n grams from academic paper corpus. The findings show that Stanford POStagger performs better than NLTK tagger for tagging accuracy and decided to use Stanford POStagger. This section outlines tokenization and part-of-speech tagging techniques.

Text Tokenization

According to Wikipedia's definition, tokenization technique, when used in lexical analysis, refers to the process of breaking a stream of text into individual words, phrases, symbols or other meaningful elements (21). These elements are known as tokens. There is no specific definition for "tokens". Many studies break texts down into individual English words. In languages that use inter-word spaces, tokenizes usually separate tokens by whitespace characters like punctuations, line break or spaces. In languages that have no word boundaries, tokenisation is very difficult. However, this study only considers English articles.

As mentioned above, tokenisers use spaces, punctuation and line breaks to separate tokens. This is trivial task. However, certain challenges exist due to variations in natural language. The first challenge comes from abbreviations. In English, abbreviations are followed by periods. For instance, we have the following sentence:

Dr. Wang is traveling to the U.S. to attend a meeting.

Dr. and *U.S both have periods*. Obviously, we should not regard the periods in these abbreviations as token separators. A straightforward commonly used way to get around this problem is to maintain a list of known abbreviations and when a word on the list is encountered, the word is directly tokenised as a single token. The quality of the list directly influences the accuracy of tokenization. Anyway, it is impossible to recognize all of the abbreviations in natural language text. Also problematic are numeric and special expressions such as URLs, citations and formulas. Special expressions may incur complex situations and there is no specific solution to this problem. Some research works may pre-process the text to perform normalization.

This research uses the word tokeniser provided by Python Natural Language Toolkit (NLTK). NLTK has been introduced briefly before. NLTK word tokeniser breaks text into individual words based on traditional white space separators –punctuations and spaces. NLTK

word tokeniser can recognise the periods in abbreviations and maintain the abbreviation as a token. Suppose that we have following sentence:

Dr.Wu works in the U.S., we are good friends.

The returned list of tokens is ['Dr.', 'Wu', 'works', 'in', 'the', 'U.S.', ',', 'we', 'are', 'good', 'friends', '.'], where *Dr.* and *U.S.*maintain their original form. Note that we aim to extract noun grams as candidate entities, in which numbers and punctuations are unusual. Therefore, special expressions such as numbers or formulas are not considered and are filtered.

Part-of-speech Tagging

Part-of-speech tagging is a process whereby a part-of-speech is assigned to each word in a text. Part-of-speech tagging plays a crucial in linguistic analysis, information retrieval and data mining. It is often used as the pre-step of text parsing.

Part-of-speech tagging is difficult considering the ambiguous role of words in natural language text. First, we can take a look at the following sentence:

Plants need water and light.

In this sentence, *plants* is plural noun. However, in the following sentence:

Each of us plants a tree.

Here, *plants* is a verb. This shows that the same word may have different part-of-speech depending on its use in different sentences. This ambiguity requires the tagging system to consider the structure and content of a text, rather than only mapping the word to a fixed tag. Currently, there are three major part-of-speech tagging solutions.

The first is called Rule-Based POS tagging, which is also the oldest approach for POS tagging. Rule-Based POS tagging uses hand-written rules. It uses dictionaries and word banks to select possible tags for words. When it comes to part-of-speech ambiguity, the Rule-Based

method also uses hand-written rules to decide which part-of-speech tagger should be chosen. The preceding word, the following word, the linguistics characteristics of the to-be-tagged word are all good sources for eliminating ambiguity.

The second type is Transformation-based tagging, which was created by Eric Brill in the 1990's (22). Brill's tagger is a kind of supervised-learning approach that aims to minimise tagging errors. The basic process of Brill's tagger is assigning a tag to each word in the text and then changing the tag according to some rules until the errors are smaller than a threshold. The algorithm maintains a dictionary that includes the most frequent tags of some words. The algorithm first scans the list to initially tag the words. For unknown words that are not in the list, the initial tag may be decided by analysis of the prefix, suffix or capitalization of the word. Brill's tagger than applies rules to check and change the tags of the incorrectly tagged word interactively. With these iterative application of the rules, the accuracy of the tagging process increases. Once the threshold is reached, the process terminates and an accurate part-of-speech tagging is achieved.

The third type is stochastic tagging. Stochastic tagging refers to any type of approach that uses probability or frequency. A large training set is necessary. The simplest stochastic tagging is to assign the most frequently occurring tag of a specific word in the training corpus to an instance of that word in the text to be tagged. This approach has a setback –– an unreasonable sequence of tags may be assigned as it only considers individual words. Another alternative approach is to calculate the probability of the occurrence of a sequence of tags. Considering that the part-of-speech of a word in a sentence is strongly related to the preceding and subsequent words, this approach makes more sense. In order to efficiently search the most possible tag of a word given previous tags, many systems use Viterbi Algorithm, which is a "trimming" approach in breadth-first search based on Maximum Likelihood Estimates. The Hidden Markov Model approach uses both word frequency and tag-sequence frequency. While this algorithm is very efficient, it is

reliant on the output of tag sequence and cannot be implemented in an automatic strategy. One solution is to use the word information instead of tag information to construct new sequences.

This research uses the Stanford POStagger, which is a maximum entropy-based tagger.

Gram Validation Check

The extracted n gram lists from the process outlined above are not what was expected due to the presence of some meaningless words. These words are obviously not semantic scholarly entities but they exist in almost every candidate entity list. They may be present for several reasons. First, the POS tagger is not 100 percent reliable. Extracted n grams may include unexpected words, letters or symbols. Prepositions such as “from”, “between”, error-encoded characters, single letters, symbols and even punctuations may appear in some grams. Second, certain adjectives may make it impossible for the gram to become semantic scholarly entities. Based on our observations, grams beginning with “many”, “same”, “only”, “correct” and some other adjectives could not be labeled as semantic scholarly entities. Several rules for candidate entities were devised based on our observations, to reduce the effort of labeling SSEs:

1 Any candidate gram should not contain punctuation marks except the joiner “-”.

2 Any candidate gram should not contain error-encoded characters. All the characters should be encoded in Unicode and be recognizable to a human.3 The first word of a gram should not be a single lower-case single letter.

4 There should not be more than two single letters in a gram.

5 The first letter of any unigrams must be in upper-case.

6 Any unigrams should not consist of only one lower-case or upper-case letter.

7 Any unigrams should not begin with a number followed by a lower-case letter.

7 Any candidate gram should not begin with a word that is included in our leading word list, which is shown in Appendix I.

8 Any candidate gram should not contain a word that is included in our word list shown in Appendix II.

The extracted grams that do not meet the above rules are automatically eliminated. This strategy effectively reduces the number of candidate entities to be labeled in the training data. Labeling is very tedious and time-consuming work.

Features

In supervised machine-learning models, each sample has a feature vector that contains a specific amount of feature values. The feature values of all the samples construct the feature space for building the machine-learning models. This study uses a total of 13 features. This section will introduce the features in detail.

Term Frequency

Term frequency (TF) is the number of times a n grams appears in a file. TF is strongly related with Term Weighting (Hans Peter Luhn, 1957). According to Wikipedia, term weighting is proportional to the value of TF of this term (23). Therefore, TF to some extent reflects the importance of a word in a file, which is a criterion of choosing a scholarly entity.

For instance, in the paper *Using Corpus-derived Name Lists for Named Entity Recognition*, the term "F-measure" appears 16 times with a TF value of 16. It is a semantic scholarly entity because it refers to the widelyused measure method in statistics.

However, it should be noted that certain grams with fewer occurrences may also be important scholar entities. For this reason, the DF and TFIDF features are added, which will be discussed later. This thesis uses the raw form of TF – the number of times a n gram appears in a file.

Document Frequency

Document frequency (DF) is the number of documents in the document corpus in which a specific n gram occurs. Certain words may occur many times in the corpus. For instance, the noun "paper" probably occurs in every document, but "paper" is not a scholarly entity at all. Therefore, a high DF value may reduce the possibility of a gram being a scholarly entity. This feature can differentiate common but unimportant grams from crucial grams.

Term frequency–inverse document frequency

Term frequency – inverse document frequency (TFIDF) is the product of term frequency and inverse document frequency, where inverse document frequency equals to dividing the number of documents in the corpus by the DF of the gram. According Patrice and Laurent, TFIDF measures the degree of the gram to a document in the document corpus (24). Wikipedia states that the TFIDF value increases with the number of occurrences of a gram in one document, while it is offset by the number of documents in which this gram appears (23). This feature can eliminate certain frequently appearing but unimportant grams such as "paper", "conference", "Reference", etc. In this program, TFIDF is defined as formula (3.1)

$$\mathrm{TF}IDF(gram, D) = (1 + \log_2(TF(gram, D))) \cdot \log_2(\frac{N}{DF(gram)}) \tag{3.1}$$

Here, D represents a document in the corpus, N is the number of the documents in the corpus. TF and DF is the tf and df value of the gram.

First letter capitalised

The feature value equals 1 if any word in the n gram starts with a capital letter, or 0 if not. This feature is created based on the observation that some scholarly entities have a capitalised first letter. For instance, names always start with a capitalised letter and it is possible that algorithms named after their creators appear in research papers, constituting important scholarly entities. It should be noted that candidate grams may consist of up to 5 words. For a gram with more than one word, the first letter in the first word may be capitalised due to the position of this word in a sentence. Therefore, in the program I check the second to last words in a k-gram where k is bigger than 1.

All letters capitalised

This feature equals to 1 if the gram includes a word where all of its letters are capitalised, or 0 if not. This feature is based on the observation that scholar entities may consist of capitalised letters. Abbreviations are usually composed of capitalised letters and they are a good source of scholarly entity. For instance, "SVM", which is the abbreviation of *Support Vector Machine*, is a scholar entity in the field of machine learning. "IEEE" is the abbreviation of a world-wide organisation *Institute of Electrical and Electronics Engineers*. For grams that consist of more than one word, the program checks the second to the last word. If there is a word that is entirely capitalised, the feature value is 1.

Containing citation

This feature equals to 1 if there is a citation after or in front of the gram in a specific range in the document text, or 0 if not. This feature is created as sentences containing citations are more likely to have scholarly entities. For instance, in the paper *The Efficiency of Multimodal Interaction for a Map-based Task* by Philip, David and Josh, there is a sentence:

> *Quick Set confirms its interpretation of the user input after multimodal integration [11], thereby allowing…….*

In this sentence, there is a citation symbol "[11]". A 2-gram "multimodel integration" before this citation is a scholarly entity. The following sentence from *Machine Translation of Very Close Languages* by Jan Haji provides a further example:

> *The work on the Czech-to-Russian MT system RUSLAN (cf. Oliva (1989)) started in 1985.*

In this sentence, a scholar entity "RUSLAN" (a translation system) appears before a citation. This program checks the previous and following 200 characters of the gram using regular expression.

Pointwise Mutual Information

Pointwise mutual information (PMI) is a form of association measurement in collection extraction. Gerlof Bouma argues: "PMI is a measure of how much the actual probability of a particular co-occurrence of events p(x,y) differs from what we would expect it to be on the basis of the probabilities of the individual events and the assumption of independence p(x)p(y)"(25).For a two-gram which consists of two words x and y, PMI is defined as:

$$PMI(x;y) = \log \frac{P(x,y)}{P(x) \cdot p(y)} \tag{3.2}$$

where P(x,y) denotes the frequency of occurrence of the two-gram in the whole document corpus, and P(x) and P(y) represent the frequency of the words x and y in the corpus respectively. The frequency of occurrence of a word is calculated by dividing the number of occurrences by the total number of words in the corpus. As we consider n grams, where n varies from 1 to 5, to be candidate scholar entities, we must apply the chain-rule of PMI. According to Wikipedia, when the number of words in a gram is bigger than 2, PMI can be computed according to the chain-rule (26). For instance, formula 3.3 illustrates how to apply chain-rule for calculating the PMI of a 3-gram:

$$\begin{aligned} pmi(x; yz) &= pmi(x; y) + pmi(x; z \mid y) \\ &= \log \frac{p(x, y)}{p(x) \cdot p(y)} + \log \frac{p(x, z \mid y)}{p(x \mid y) \cdot p(z \mid y)} \\ &= \log \frac{p(x, yz)}{p(x) \cdot p(yz)} \end{aligned} \tag{3.3}$$

In the formula, x, y and z are three words that consist of the three-gram ‘xyz’. Thus, we can use the frequency of one-gram ‘x’ and two-gram ‘yz’ to compute the pmi of three-gram ‘xyz’. The same rule can also be applied to compute the four and five grams.

Here we can provide some examples.

A two-gram “machine learning” can be computed as follows:

$$pmi("machine\_learning") = \log \frac{p("machine\_learning")}{p("machine") \cdot p("learning")}$$

where p(word) denotes the frequency of the word. Similarly, the three-gram “support vector machine” can be computed as

$$pmi\left("support\ vector\ machine"\right) = \log \frac{p\left("support\ vector\ machine"\right)}{p\left("support\ "\right) \cdot p\left("vector\ machine"\right)}$$

Some candidate grams are random combination of words, while others are frequently appearing combinations, which is a good indicator of a possible scholar entity. As mentioned

above, PMI measures the degree of association of the words in a gram. Therefore, we use PMI as a feature for the classification model.

Upper case and lower case mixed

Some terms are combinations of upper case and lower case letters. This feature equals to 1 if any word in the gram is a combination of upper and lower case letters, or 0 if not. An example is“PTMiner”, which appears frequently in the paper entitled “Automatic construction of parallel English-Chinese corpus for cross-language information retrieval”, which is a parallel-text searching system. A combination of upper and lower case letters in one word could be a sign of a name of a specific system, place, company, etc, which merits treating this characteristic as a feature. It should be noted that just the second to the last letter of each word are checked as the first letter should be in upper case where the word appears in the beginning of a new sentence. For n grams that consist of several words, we check each word and return 1 if anyone is a combination.

Position of the n gram

Academic papers are usually composed of common sections including a title, abstract, introduction, body and references. Different parts play different roles. For instance, readers read theabstract to gain a brief overview of the major themes of the paper, and readers find relevant articles in references. This results in varying probability of occurrence of important words or phrases from different parts of the document, and therefore different possibility of occurrence of scholarly entities. We argue that the authors of papers tend to put important scholarly entities related to the paper’s content in the title and abstract to provide readers with a concise overview.

Thus, we consider the existence of a candidate scholarly entity in different parts of the paper as a feature of itself. This project has three key sections: title, abstract and the paper body.

(1) Title

Readers read titles first and gain an overview ofwhat the paper is about. Thus, the noun phrases in the titles are much more likely to be content-related, increasing their likelihood of containing scholarly entities.

(2) Abstract

An abstract is a short and powerful statement of the purpose, contribution and results of the paper. Readers read abstracts to decide whether the paper is interesting or not. Therefore, scholarly entities that are highlyrelated to the paper tend to be present in the abstract.

(3) Body

The body of an academic paper includes all the detailed contents. The body may include a problem introduction, related works, approach descriptions, results and future works. As the body is the main section of a research paper, most of the candidate scholarly entities come from this section. Scholarly entities related to specific techniques are likely to occur in the body of the paper.

I use three individual Booleans to represent this feature. Each of them corresponds to a title, abstract and body respectively. If the gram occurs in the above four parts, the value in corresponding position equals to 1, otherwise it is 0. For instance, if a gram occurs in the abstract and the body, the values should be (0, 1, 1).

Contained in Brackets

This is a feature with Boolean values. This feature equals to 1 if a unigram is surrounded by brackets in the original text of the paper. It equals 0 if it is not a unigram or the unigram is not in brackets. This feature was created as abbreviations of lengthy scholarly entities are at times put in brackets when they first appear in a paper. An example from *Language-Integrated Querying of XML Data in SQL Server(James F. Terwilliger, Sergey.Melnik, Phil.Bernstein)* is shown below:

> *Several object-relational mapping (ORM) frameworks [2] have emerged to help application developers bridge objects and relations.*

In this sentence, *ORM,* which is put in a bracket, is the abbreviated form of *object-relational mapping.* According to our definition, ORM is a semantic scholarly entity.

Contained in Quotations

This is a feature with Boolean values. This feature equals to 1 if a gram is surrounded by quotes in the original text of the paper, or 0 where it is not. Words or phrases are placed in quotation marks for emphasis. This feature has been created as an author's emphasis on certain words or phrases may indicate the importance or special usage of such words or phrases. The following sentence from *Detecting Semantic Cloaking on the Web,* is a good example:

> *"Nepotistic links", i.e., those links between pages that are present for reasons other than merit, are bad for the linkbased ranking algorithm, so it is necessary to get rid of them.*

The 2 gram *Nepotistic link*sis placed in quotations for emphasis. Here, *Nepotistic links* represents a specific kind of network link and it is a semantic scholarly entity.

Chapter 5

Training Sample Labeling

As mentioned above, our project aims to build a supervised binary classification model. In supervised learning, we must provide classified data as the training set to the classifier. Noun grams have already been extracted from scientific papers and their feature vectors calculated. Now the training data should be constructed by classifying some of the candidate semantic scholarly entities. In this section, we introduce the basic rules, mechanisms and examples of SSE labeling.

Labeling procedure

In our project, we manually label semantic scholarly entities. For each chosen file to be labeled, we examine and study each candidate SSE in the list, and decide whether a candidate SSE is or is not a real SSE based on the rules, the semantic meaning of the phrase and our domain knowledge. To guarantee the validity and effectiveness of labeling, we created a procedure in which three people are involved in the labeling of candidate SSEs. Fig.4.1 is the flow chart of the procedure. In this procedure, 2 to 3 people are enrolled in the labeling to avoid personal bias. We choose several files, which correspond to different papers in different conferences or journals. Initially, person A and B label these files independently. Then person A and B together check the labeling of each candidate SSE in these files. Where they cannot reach a consensus on a candidate SSE, this SSE will be delivered to person C to label. Finally, A, B and C will discuss these controversial labelings until they reach an agreement.

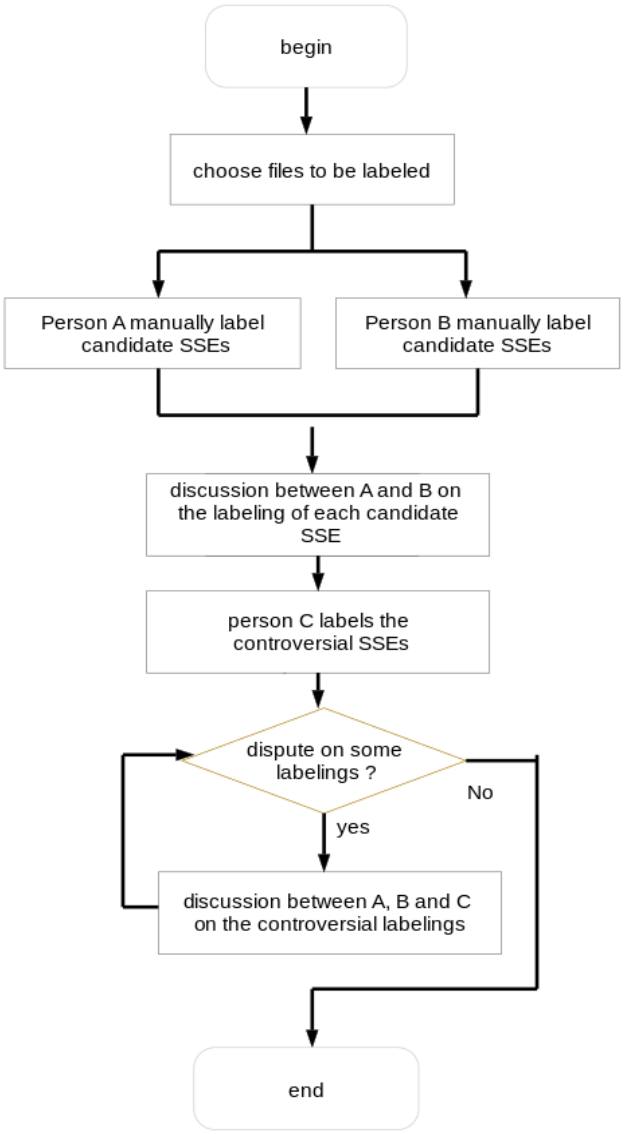

Figure 5.1 Labeling process flow chart

File Selection

Our scientific papers come from three world-popular computer science – related conference and journals, which are listed below:

- Association for Computational Linguistics, ACL
- Neural Information Processing Systems, NIPS
- Very Large Date Bases, VLDB
- International World Wide Web Conference, WWW

To prevent bias toward a specific conference, at least one file has been selected from each of the conferences of journals. The information of the corresponding papers of the chosen files and their origins are listed in table 4.1:

Table 5.1 Conference information of the chosen papers

Paper name	Conference name
Position Paper: Towards the notion of gloss, and the adoption of linguistic resources in formal ontology engineering	WWW
Detecting Semantic Cloaking on the Web	WWW
Language-Integrated Querying of XML Data in SQL Server	VLDB
Semandaq: A Data Quality System Based on Conditional Functional Dependencies	VLDB
A Data Warehousing Architecture for Enabling Service Provisioning Process	VLDB
Using Corpus-derived Name Lists for Named Entity Recognition	ACL
Using Context Inference to Improve Sentence Ordering for Multi-document Summarization	ACL
Regularized Distance Metric Learning: Theory and Algorithm	NIPS
CollaborativeRankingWith17Parameters	NIPS
Tensor Decomposition for Fast Parsing with Latent-Variable PCFGs	NIPS

Labeling rules

Semantic scholarly entity is a new concept. This study creates a new set of rules to determine whether a candidate semantic scholarly entity may be labeled positive or not. Basically, semantic scholarly entities are nouns that belong to the following six types:

- Tools.

  Tools may include software and computer languages. Some examples are “Java”, “SQL”, “MATLAB”.

- Methods

  Methods may include algorithms and theorems. Some examples are “Relevant Component Analysis”, “conjugate gradient method” and “divide and conquer algorithms”.

- Concepts

  Most of the positive-labeled candidate entities are concepts. Concepts include any terms that describe an idea in computer science. Examples of semantic scholarly entities that are regarded as *concepts* include“XML”, “machine learning”, “training data” and “automatic speech recognition”.

- Dataset

  This type includes widely-used datasets such as “UCI datasets” and “WEBKB”.

- Process

  Example: “normalization”.

- Parameter

  Example: “penalty parameter”.

Whether a candidate entity belongs to the above three types is the basic rule in the labeling procedure. However, there are some other issues. First, we are looking for semantic scholarly entities, in which *semantic* indicates that the original text is useful in labeling. Certain English words have different meanings in different context, leading to ambiguity when labeling some of the entities. Where there is ambiguity, we first search the entity to be labeled in the original text and consider the semantic meaning of the entity in the text based on our domain knowledge. For example, *NE* is a unigram in the candidate list of the paper *Using Corpus-derived Name Lists for Named Entity Recognition*. We cannot decide whether we should label it as a real SSE merely based on its spelling, as it is possible that this unigram comes from a formula. However, we found the first appearance of this unigram:

> *These results provide a baseline against which the con- tribution of more sophisticated supervised learning techniques for NE recognition should be measured.*

In the sentence, NE semantically refers to a type of recognition target terms (named entity), which is a scholarly entity.

Second, we avoid giving positive labels to the candidate multi-word entities that are self-explanatory and do not have any scholarly meanings. While some candidate entities contain positive-labeled SSEs, they are easy to understand and are not scholarly. One example of such an entity is “Linux operation”, which is extracted from a paper from NIPS. “Linux” is a scholarly entity, but “Linux operation” cannot be labeled as a real semantic scholarly entity as it does not refer to a scholarly concept. No one would ask questions such as “what is Linux operation” or “can you explain Linux operation”. As such, these entities are meaningless in our semantic scholarly entity extraction task.

Third, well-known companies, places and organisation names are not of interest.Nor are people’s names relevant either. We do not regard them as semantic scholarly entities because they do not refer to any domain knowledge. On the other hand, most of these names can be found in

Wikipedia with detailed information. Our project does not need to extract these entities from scientific papers. In addition, may previous studies have already been conducted on how to recognise specific kinds of names currently, so they do not need to be included as entities in this system.

Examples

This section provides some examples of real semantic scholarly entities that we labeled. The examples are shown in table 4.2:

Table 5.2 Examples of positive semantic scholarly entities

Type	Unigrams	Multi-grams
Tools	Java, SQL, MATLAB	Linux system, SQL query
Methods	LDA	RelevanceComponentAnalysis, online learning algorithm, part-of-speech tagger
Concepts	XML, DB, DataInequality	Cross validation, normal distributions, entity recognition

Chapter 6 Sampling algorithms

Our data is imbalanced and sampling strategies are necessary. In this chapter, the three sampling strategies used in this project are introduced– simple over-sampling, simple under-sampling and Synthetic Minority Oversampling Technique.

Sampling from imbalanced data

In our project, each paper may have several hundred candidate semantic scholarly entities. However, only a small number of them will be labeled as positive semantic scholarly entities. Based on our statistics, the ratio of semantic scholarly entities to normal phrases in a paper is 1 to 3, indicating that our data is imbalanced.

Making use of imbalanced data for machine learning may lead to an imbalanced degree of accuracy. Most testing samples would be classified as the majority class, causing low recall of the machine-learning model. This project aims to identify semantic scholarly entities in scientific papers and a low recall would make our model unreliable or even useless. Thus, we need to implement some strategies to reduce the influence of the imbalance of the dataset.

Sampling is a popular strategy in this situation. According to Haibo He and Edwardo A. Garcia (3), sampling strategies in machine-learning field on imbalanced data can modify the original dataset using certain mechanisms to provide a more balanced data distribution. They argue that for most of the imbalanced dataset, the sampling strategy does help to improve the performance of the ML model. In our project, we implement three different kinds of sampling mechanisms – random oversampling, random undersampling and synthetic minority oversampling techniques. These mechanisms are described in the following paragraphs.

Random oversampling, as its name suggests, is a sampling mechanism that works by adding duplicate samples chosen randomly from the minority class to the dataset. Suppose that we have collection of samples S. To implement random oversampling, we randomly choose an amount of E samples from the minority class in S to generate a sample collection S_{min} and add S_{min} to S to create a new training data set. Usually, E equals the difference between the number of samples in the majority and minority class in S to guarantee that a relatively equal balance of majority and minority class samples in the new data set. However, we can increase or decrease E to vary the degree of class distribution balance.

Similarly, random under-sampling removes randomly selected samples from the majority class. Take for example a collection of samples S. In random under-sampling, E majority class samples are randomly selected and removed from S. Usually, E equals to the difference between the number of majority class samples and minority class samples to guarantee that the new dataset is balanced.

While random oversampling and under-sampling are easy to understand and implement, they may hinder the learning process. Haibo and Edwardo argue, that in under-sampling, removing samples may cause the classifier to lose important concepts pertaining the majority class. In the case of oversampling, problems can arise from duplicate samples. Multiple instances of the same sample may result in over-fitting of the classifier, leading to poor performance when predicting new data. To overcome these drawbacks, people create several relative complex sampling techniques. This project implements a synthetic minority oversampling technique (SMOTE).

SMOTE is a powerful sampling technique that has shown success several times in application. Unlike random oversampling, SMOTE generates new data points instead of replicating data points. It creates artificial data based on the similarities between the data points of

the minority class. Suppose that we have a sample collection S and the minority class sample $S_{\min} \in S$. In SMOTE, for some integer K, we find the K-nearest neighbors of each data point $x_i \in S_{\min}$. To create a synthetic data point, we first randomly select one of the K-nearest neighbors of an existing point, and then multiply the difference of each feature in the feature vector between x_i and its neighbor with a random number between 0 and 1. For example, where $\hat{x}_i$ is one of the nearest neighbors of x_i, a synthetic data point can be calculated as follows:

$$x_{syn} = x_i + (\hat{x}_i - x_i) \cdot \delta$$

where $(\hat{x}_i - x_i)$ represents the corresponding difference of each feature values in the feature vector of $\hat{x}_i$ and x_i, $\delta \in [0,1]$. This process creates a data point located on the line that joins x_i and one of its neighbors. While SMOTE has certain drawbacks with over-generalisation and variance, it overcomes the problems generated by simply replicating data and helps to improve the performance model learned from imbalanced data.

This research compares the performance of these three sampling methods on different machine learning models. We also compared the performance of the same machine learning models with and without sampling.

Chapter 7

Machine Learning Model Building and Evaluation

We use Support Vector Machine, Decision Tree and Random Forests to build machine-learning models.To balance positive and negative training samples, we use simple under-sampling and SMOTE (Haibo He and Edwardo A. Garcia, 2009). This section describes how the models are trained using our manually labeled data and evaluation.

Data preparation and evaluation strategies

We described the process of data labeling in chapter 5. In total, 10 academic papers were labeled with 4617 grams, including 835 positive labels (candidate entities labeled as semantic scholarly entities) and 3782 negative labels. In order to reduce the inference of over-fitting and to increase the validity of the evaluation, we first choose 500 n gram as the testing dataset. The remaining 4117 n grams are used for model training and parameter searching.

Three sampling strategies are implemented for each model. On the other hand, we implement a grid search strategy to find the best parameters combination in each situation based on classification accuracy. That is, all combinations of parameters are exhaustively considered within a specified range, and the ones that show the highest classification accuracy are selected. The search range is based on whether there is a peak value of classification accuracy. When training models and searching for the best parameters, we use 5-fold cross validation.

Obtaining the trained best classifier model, we use precision-recall curves to evaluate the classifiers. In information retrieval, precision represents the fraction of retrieved instances that are

relevant, while recall is the fraction of relevant instances that are retrieved. Changing the classifier threshold changes the precision and recall accordingly. This property provides us with a curve called a precision-recall curve. A precision-recall curve is often used in binary classification problems where data is imbalanced. As this study aims to extract semantic scholarly entities from computer science papers, it is more important to successfully extract the semantic scholarly entities and whether the extracted entities are accurate. The area under the precision-recall curve (AUC) is a measurement of the performance of the classifier. Actually, this area is equivalent to the mean precision of the classifier with respect to different thresholds. The greater the area is, the better the classifier.

Logistic Regression Machine-learning Model

Parameter search and model training

Table 8.1 illustrates the best parameters chosen with their classification accuracy for simple under-sampling, simple over-sampling and SMOTE. We use the 4117 n grams for training, and the accuracy values are acquired during the training process.

Table 7.1 Best parameter combination of Logistic Regression

Sampling Strategy	Regularisation strength C	Solver	Accuracy
Under-sampling	0.01	lgfbs	0.584144645341
Over-sampling	1.0	newton-cg	0.590909090909
SMOTE	1000.0	lgfbs	0.591319052988

Table 8.1 shows that the accuracy values of under-sampling, over-sampling and SMOTE are similar. The accuracy value of the under-sampling model is slightly lower than the other two. This is because the number of n grams in the training data of the under-sampling is smaller.

Model Evaluation

Figure 8.1 shows the precision-recall curves of the three Logistic Regression classifiers. Table 8.2 displays the precision, recall, accuracy and AUC of the three classifiers. It should be noted that the 500 n grams test dataset is used to plot the curves and calculate the statistics in the table.

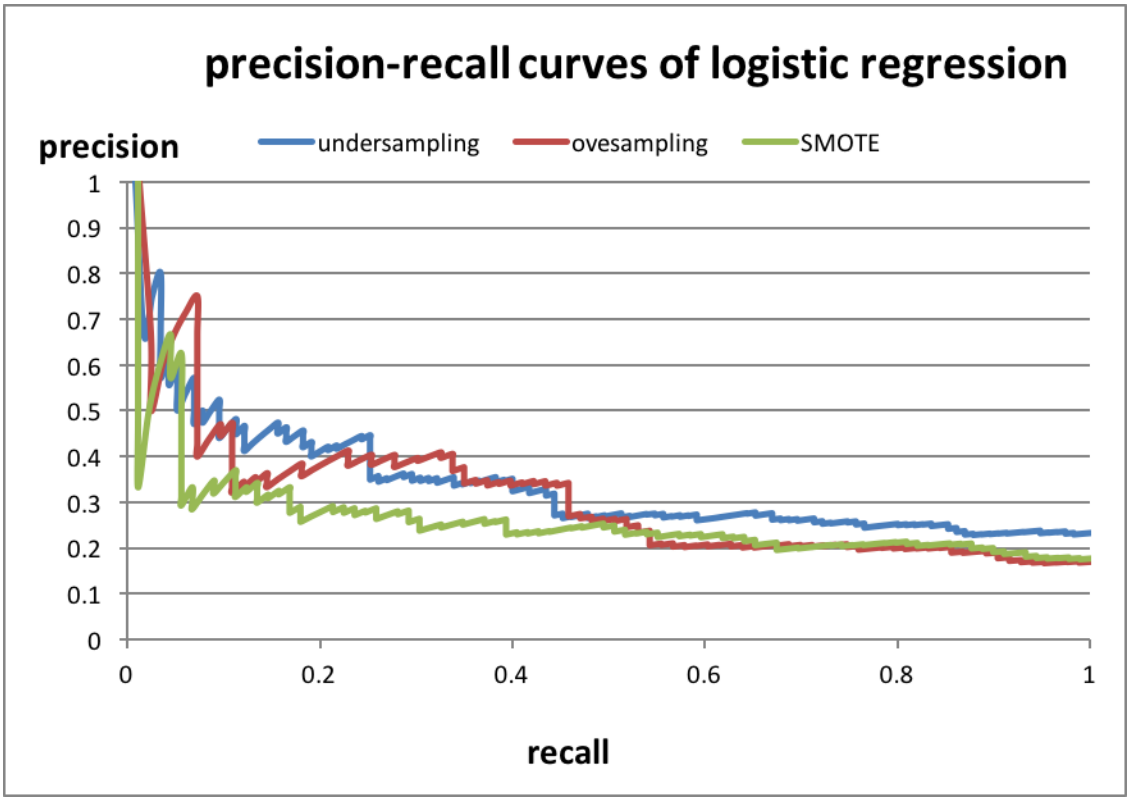

Figure 7.1 Precision-recall curves of Logistic Regression

Table 7.2 Statistics of the three Logistic Regression models using test dataset

Sampling strategy	precision	recall	accuracy	F-measure
Under-sampling	0.34	0.34	0.70	0.340
Over-sampling	0.25	0.52	0.66	0.337
SMOTE sampling	0.23	0.58	0.57	0.329

From Figure 7.1 shows that the three curves share similar trends. When the precision is high, the recall values are low. With the increase of the recall, precision decreases dramatically at first, and then it becomes stable when the recall values are higher than 0.5. In Table 7.2 we can see that the model with under-sampling strategy has the highest precision and accuracy, but its recall is the lowest among the three models. SMOTE-sampling and over-sampling models are similar; they have higher recalls but lower precisions and accuracies. In terms of the F-measure, the under-sampling model is the best. Thus, we think under-sampling performs the best in Logistic Regression model.

Support Vector Machine model

Parameter search and model training

This project applies the widely used radial basis function (RBF) kernel described in the last chapter. In SVM model, there are two control parameters, namely C, and γ. Penalty parameter C controls the cost of misclassification. A large C means a high cost of misclassification, forcing the model to explain the training data "stricter". A large C results in low bias and high variance, making the model more prone to over-fitting. Conversely, γ controls the influence of support vectors. A large γ results in high bias and low variance. Thus, γ and C can trade off each other. To complete the grid search in a reasonable time, both C and γ will change

by the power of 10. A 5-fold cross validation is adopted in each training of a specific combination of parameters.

Table 8.1 illustrates the best parameters chosen with their classification accuracy for simple under-sampling, simple over-sampling and SMOTE. We use the 4117 n grams for training, the accuracy values are acquired during the training process.

Table 7.3 Best parameter combination of SVM

Sampling Strategy	Parameter C	Parameter γ	Accuracy
Under-sampling	10.0	1.0	0.67
Over-sampling	100000.0	1000.0	0.87
SMOTE	100.0	10.0	0.75

Table 8.1 shows that the highest accuracy values of SVM classifier using simple over-sampling and SMOTE are higher than the accuracy of the classifier using under-sampling. However, it should be noted that this is "training" accuracy, which is calculated using training data and cross validation strategy. The under-sampling based model fares worse as it eliminates many negative n grams, leaving a small training dataset, which adversely impacts its performance. On the other hand, SVM with over-sampling has the highest "training" accuracy, but this may result from over-fitting.

Model Evaluation

Figure 8.1 shows the precision-recall curves of the three SVM classifiers. Table 8.2 summarizes the precision, recall, accuracy and AUC of the three classifiers. Note that the 500 n grams test dataset is used to plot the curves and calculate the statistics in the table.

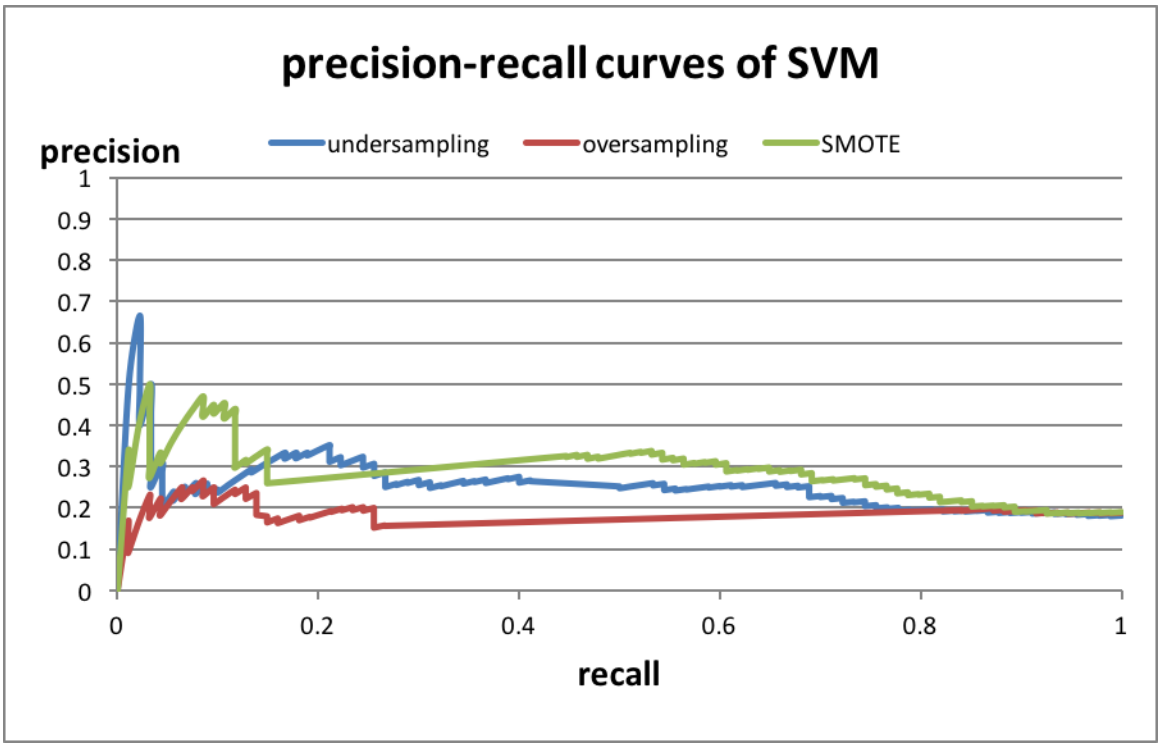

Figure 7.2 Precision-recall curves of SVM

Table 7.4 Statistics of the three SVM models using test dataset

Sampling strategy	precision	recall	accuracy	F-measure
Under-sampling	0.25	0.62	0.6	0.36
Over-sampling	0.23	0.11	0.76	0.15
SMOTE sampling	0.32	0.56	0.69	0.41

Figure 7.2 shows that the three curves have similar patterns. When the recall is higher than a certain value (e.g. 0.15 for SMOTE-based SVM classifier), the precision decreases slowly from 0.3 to 0.2. However, when the recall is low, the precision is high. An increase in the recall leads to a dramatic decrease in precision. Table 7.4 shows that the SVM classifier based on under-sampling has the highest recall, the SMOTE-based classifier has the highest precision and the over-sampling based classifier has the highest accuracy. As a NER project, the precision and the recall values are more important. We think SMOTE-based SVM classifier performs the best

regarding the F-measure, and which is a combination of the precision and recall. . The over-sampling based classifier is the poorest performer.

Random Forests Model

Parameter search and model training

Random forests is an ensemble learning algorithm based on decision trees. It is necessary to search for at least the best combination of the two parameters - max_depth and max_features, of the Decision Tree classifier. Another parameter for Random Forests algorithm is the number of decision trees used– n_trees. Again, grid search is used to identify the best combination of the three parameters based on classification accuracy. In each training of a specific combination of parameters, we adopt 5-fold cross validation.

Table 7.6 shows the best parameters based on their classification accuracy for simple under-sampling, simple over-sampling and SMOTE.

Table 7.6 Best parameter combination of Random Forest

Sampling Strategy	Max depth	Max features	Number of trees	Accuracy
Under-sampling	11	1	71	0.64
Over-sampling	131	2	101	0.88
SMOTE	11	6	91	0.78

Table 7.6 shows that based on classification accuracy, random forest algorithm based on over-sampling and SMOTE sampling outperform simple under-sampling. Simple oversampling acquires the best accuracy of the three. As mentioned previously, these accuracy values are calculated using training data. For similar reasons as SVM and DT, the under-sampling strategy under-performs over-sampling and SMOTE.

Model Evaluation

Figure 8.3 shows the precision-recall curves of the of the three Random Forest classifiers. 500 test data is used to draw the precision-recall curves. Table 8.6 shows the precision, recall, accuracy and AUC of the three classifiers acquired using the test dataset.

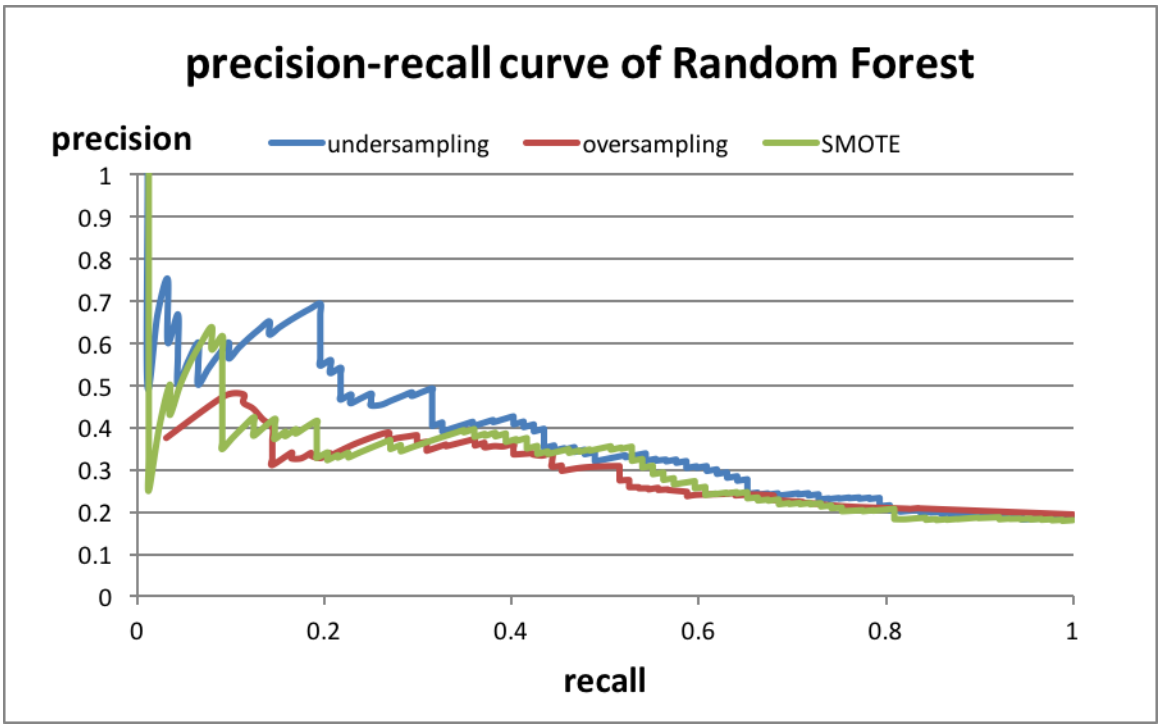

Figure 7.3 Precision-recall curves of Random Forest

Table 7.7 Area under the curve of random forest

Sampling strategy	precision	recall	accuracy	F-measure
Under-sampling	0.32	0.54	0.70	0.40
Over-sampling	0.36	0.37	0.75	0.36
SMOTE	0.38	0.38	0.778	0.38

Figure 7.3 shows that the curves of the three Random Forest classifiers are similar. On the left part of the curve, the precision values decrease and the recall increases with the change of the threshold. When the recall becomes greater than 0.4, the precision decreases more slowly.

From Table 7.7, we can see that the recall value and F-measure of the Random Forest classifier with under-sampling is much higher than the recall values of the classifiers with over-sampling and SMOTE. The latter two perform similarly. While the precision of the Random Forest classifier with under-sampling is slightly lower than those of the other two classifiers, we consider it to be the best one.

Model Comparisons

The three machine-learning models have certain common characteristics. First, models trained with over-sampling and SMOTE sampling are better than the ones trained with under-sampling strategy in terms of training accuracy. As outlined previously, this may be the result of the training dataset's size. Models using simple over-sampling typically have slightly higher accuracy values than the SMOTE. We should note that this accuracy is just "training" accuracy used for parameter optimization. This is likely to be caused by data duplication caused by the over-sampling process that may introduce over-fitting. It is best to use precision-recall curves to evaluate the classifiers. We notice that among the precision-recall curves of the SVM and Random Forests machine-learning model, the models with under-sampling strategy have the highest recall, and the models with SMOTE strategy have the highest precision, and vice versa in Logistic Regression. However, over-sampling based models are not outstanding in any classification algorithms; they perform badly in SVM. This may also be the result of over-fitting.

Some recent studies point to the low reliability and stability of AUC. We further calculate the F-measure, which is the harmonic mean of precision and recall, of the three selected best classifiers of the three classification algorithms. They are summarized in Table 8.7. The table

shows that the SVM model is slightly better, but the SVM and the Random Forest classification algorithms are similar.

Table 7.8 F-measure of the best classifiers of SVM, Decision Tree and Random Forest

Classification model	F-measure
SVM (SMOTE)	0.41
Logistic Regression (under-sampling)	0.34
Random Forest (under-sampling)	0.40

However, we cannot choose which sampling strategy to use based on its performance in practice. This is sampling because sampling may introduce bias. Once we believe one specific type of sampling strategy introduces the min bias, we should always use it. It is important to be consistent in terms of sampling strategies.

Feature Importance Analysis

Gini Impurity is used as the measure of effectiveness of a feature in classification. In decision tree or random forest models, Gini Impurity can be applied to examine the importance of the features. This measurement is known as Gini Importance. It is defined as the total decrease in node impurity and is weighted by the probability of reaching that node. Scikit-Learn can calculate Gini Importance while training decision tree or random forest models. We calculate the Gini Importance of each feature when training the Random Forest classifier with SMOTE. The result is summarized in figure 8.7.

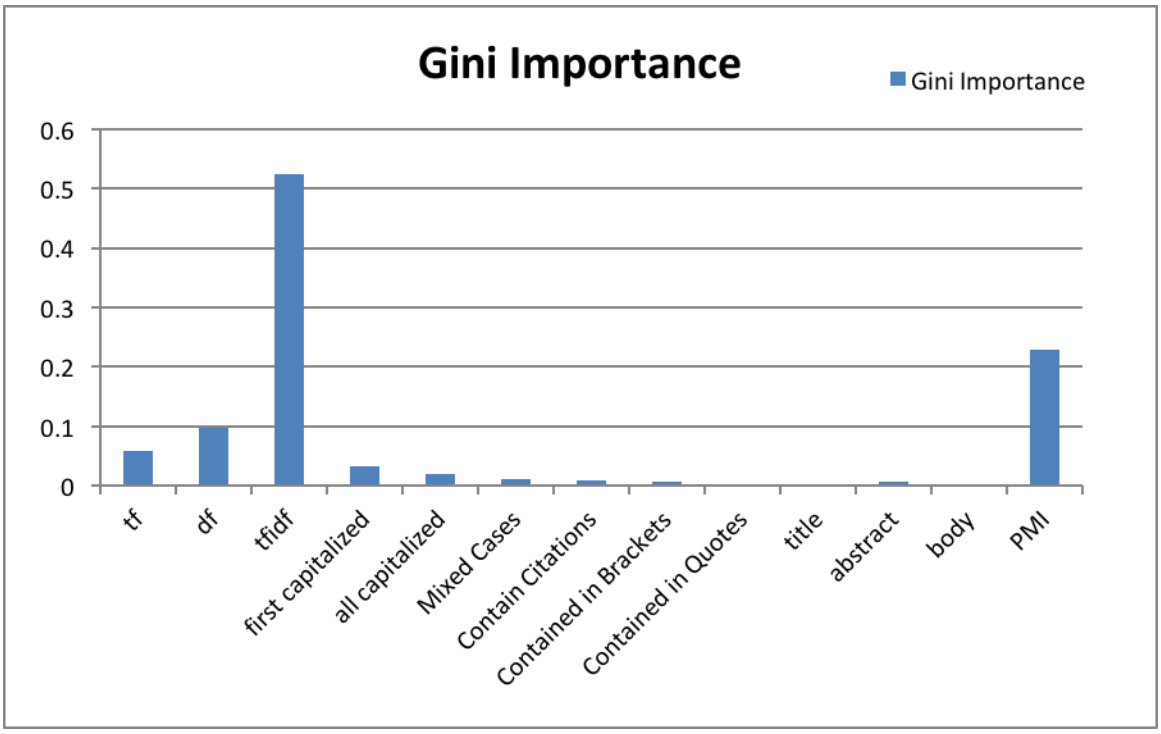

Figure 7.4 Gini Importance of 13 features

The horizontal axis shows the 13 features, while the vertical axis shows the weighted Gini Importance. Note that the sum of all the Gini Importance of all the 13 features is 1. The greater the Gini Importance, the more important the feature. Figure 8.7 shows that TFIDF plays a very important role in the decision tree classifier training process, counting for more than 50% importance. The Gini importance of PMI is the second highest – contributing more than 20 % importance. A further two numeric features, DF and TF are also more important than the other Booleanfeatures in the classifying samples. Among the 9 Boolean features, the ones that related to the case of the letters (first capitalised and all capitalised) are higher than the rest, which may indicate that uppercase letters could be a useful sign in classifying SSEs.

Extracted Semantic scholarly Entities

This section provides several examples of Semantic Scholarly Entities extracted by our algorithm. We use the paper *Towards Efficient Named-Entity Rule Induction for Customizability* (AjayNagesh et al, 2012) and Decision Tree classifier with SMOTE sampling for this demo.Table 7.9 shows the Semantic Scholarly Entities extracted from the paper.The column on the left shows all of the extracted terms that we think meet the definition of SSEs, and the two right-hand columns list the wronglyrecognised terms.

Table 7.9 Semantic Scholarly Entity Examples from an academic paper

Semantic Scholarly Entities	Wrong terms recognised as Semantic Scholarly Entity	
regular expressions	search space	interesting directions for
Rule-based systems	major challenge	future work
SystemT	empirical evaluation	E1
rule-based systems	Annotation	Tjong Kim
definite clauses	complexity measure	induced rules
IE	CD views	E2
BF	overall F measure	CoNLL03 test dataset
GATE	LOC	test dataset
CoNLL03 test collection	manageable complexity	promising results
algebraic framework	levels of abstraction	entity mention
iterative clustering	detailed analysis	Aleph
regular expression	process of building	city name
Named-Entity Recognition	select statement	Target Language
rule-based NER system	first order	Refinement
ALP	second phase	main motivation
rule-based NER	avenues for improvement	Appelt
computational overhead	manual effort	initial work
Rule Induction	Overlaps	manual rule development
ACE	domain specific features	induction algorithm
relational algebra	coarse grain	main contributions
RIPPER	type of rule	select clause
non-overlapping clusters	background knowledge	Compton
NER	direction for future work	named-entity types
agglomerative clustering	set of rules	Introduction Named-entity
CR	PerFirstLast span	recognition
grammar formalism	set of spans	contextual clues
scalar function	Introduction Named-entity	orders of magnitude
induction algorithm	CoNLL03 test	IBM
rigid designators	OrgCD	F1

induction biases	basic building block	strong clues
hierarchical agglomerative clustering	Query	initial set of rules
contextual clues	Basic Features	set of parameters
custom code	increase in complexity	directions for future work
AQL rule	E3	terms of complexity
generic NER	Fig- ure	first phase
rule induction	Widmer	first order logic
propositional rule	list of persons	basic feature sets
domain customization	special emphasis	Common Pattern
induction system	finite state transducer	Background Knowledge
Common Pattern Specification Language	entity types	Reuters news
Information Extraction	CoNLL03 training	combination of clustering
AQL	sincePhase name	PER
IE system	set of basic features	Prior work
	ORG	SQL
	LocCD	Related Work
	entity type	future work
	CD	Span-View Table
	expres- sions	formal definition
	type of bias	execution engine
	Tjong Kim Sang	effect on accuracy
	PersonCandidate tuples	Automatic generation
	interesting direction for future work	textual spans
	Prince William	low precision
	BFs	Common Pattern Specification
	collection of Reuters	PersonInvalid tuples
	De Meulder	PerCD
	interesting direction	quantitative measure
	dictionar- ies	induction process
	prespecified threshold	Span-View Table Table
	Language	class label
	report results	relevant background knowledge
	Mark Twain	little attention
	Gaines	Specification Language
	system on CoNLL03	set of CD
	notion of rule	notion of ex-tractor
	FirstNameDict	complexity
	low complexity	organization names
	customization of basic features	CO
	Chiticariu	
	building blocks	
	itera- tions	
	target language	
	dual advantages	

Among this list, several terms meet the definition of Semantic Scholarly Entity. For instance, we identified “Named-Entity Recognition”, “rule-based NER systems”, “regular expressions”, “SQL and GATE”. However, there are misclassified words such as “major challenge”, “process of building” and some totally wrong words like “E1”, “E2”.

Chapter 8

Discussion and Future Research

Discussion

This thesis proposes a new type of named entity– Semantic Scholarly Entity, and an approach to recognize Semantic Scholarly Entities in computer science research papers based on supervised classification techniques. A definition of Semantic Scholarly Entity was first provided. We then outlined our approach of extracting candidate Semantic Scholarly Entities from computer science papers and introduced 13 features for building machine-learning models. Next, three machine learning algorithms – SVM, Decision Tree and Random Forest, as well as three sampling strategies, are used to develop classification models. Finally, we evaluate each of the classification models and choose the best one based on F-measure.

The findings show that the performances of SVM, Decision Tree and Random Forest algorithms are similar where appropriate sampling strategies are used. Nevertheless, we should admit that our classification models are not good enough. The best model – Decision Tree with SMOTE, only achieves an F-measure of 0.42 and a classification accuracy of 0.75. Possibly, this is because our chosen features were unable to differentiate between positive and negative samples. Therefore, I think the best way to improve the performance of the classification model is to create more effective features that can better reflect the characteristics of Semantic Scholarly Entities. A further limitation of this study is that there are candidate entities with similar meanings in the candidate list. This increases the labor of labeling.

A drawback of our supervised classification model is that we must spend lots of time on tedious n gram labeling in order to provide enough training data. Some research works in NER

attempt to use unsupervised techniques (3) (4). However, their systems can only recognize people's names, famous organisations, locations and other traditional named-entities. Our project is the first to propose the concept of Semantic Scholarly Entity, so we do not have any existing rules, templates or characteristics of them. A training dataset can only be created by manually labeling Semantic Scholarly Entity based on our domain knowledge.

Future Works

For future research, first we plan to use entity-linking techniques. Our approach could extract candidate Semantic Scholarly Entities, but could not identify the real meaning of the entities. Many English words show ambiguity, and the same word can mean completely different things. Entity linking techniques is a task to determine the identity of terms, which could help to reduce ambiguity of the candidate entities. Second, we plan to group terms with similar meanings in the candidate entity list. Terms with similar meanings but different spellings exist in our candidate list. These terms increase the labor of labeling the training data and adversely affect the document-level and corpus-level features. It is better to use some strategies to group similar terms together.

References

[1] Named-entity recognition, Wikipedia. https://en.wikipedia.org/wiki/Named-entity_recognition

[2] General Architecture for Text Engineering. https://gate.ac.uk/

[3] David Nadeau, Peter D. Turney and Stan Matwin.Unsupervised Named-Entity Recognition:Generating Gazetteers and Resolving Ambiguity. 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Québec City, Québec, Canada, June 7-9, 2006. Proceedings.

[4] Downey, Stanley Kok AnaMaria, Popescu Tal Shaked.Web-ScaleInformation Extraction in KnowItAll.Proceedings of the 13th international conference on World Wide Web, Pages 100-110.

[5] Wikipedia, Keyword extraction: https://en.wikipedia.org/wiki/Keyword_extraction

[6] Brian Lott . Survey of Keyword Extraction Techniques, 2012

[7] K.S. Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1):11–21, 1972.

[8] Y. Matsuoand M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. International Journal on Artificial Intelligence Tools, 13:2004, 2004.

[9] YukioOhsawa,NelsE.Bensonand MasahikoYachida.KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor. ADL '98 Proceedings of the Advances in Digital Libraries Conference. Page 12.

[10] G. Ercan and I. Cicekli. Using lexical chains for keyword extraction. Information Processing & Management, 43(6):1705–1714, 2007.

[11] David Nadeau and Satoshi Sekine . A survey of named entity recognition and classification. Named entities: Recognition, Classification and Use, page 3-26.

[12] Wikipedia, Rigid Designator. https://en.wikipedia.org/wiki/Rigid_designator

[13] David D. McDonald. Internal and External Evidence in the Identification and Semantic Categorization of Proper Names. 1996.

[14] Jun'ichiKazama and KentaroTorisawa.Exploiting Wikipedia as External Knowledge for Named Entity Recognition. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational.

[15] Natural Language Toolkit, http://www.nltk.org/

[16] Stanford Natural Language Processing Group, http://nlp.stanford.edu/

[17] Kristina Toutanova and Christopher D. Manning. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), page 63 – 70.

[18] Grobid.https://grobid.readthedocs.io/en/latest/Introduction

[19] Text Encoding Initiative. http://www.tei-c.org/index.xml.

[20] Scikit-Learn,http://scikit-learn.org/stable

[21] Wikipedia, Tokenization. https://en.wikipedia.org/wiki/Tokenization_(lexical_analysis)

[22] Eric Brill. A simple rule-based part of speech tagger. Proceedings of the third conference on Applied natural language processing. Pages 152 – 155.

[23] Wikipedia, tf-idf. https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[24] Patrice Lopez and Laurent Romary. HUMB:AutomaticKey Term Extraction from Scientific Articles in GROBID. Sem Eval '10, page 248-251.

[25] Gerlof Bouma.Normalized (Pointwise) Mutual Information in Collocation Extraction. Proceedings of the Biennial GSCL Conference, 2009.

[26] Wikipedia, Pointwise mutual information. https://en.wikipedia.org/wiki/Pointwise_mutual_information#Chain-rule_for_pmi

Appendix

Illegal start word list

'same', 'similar', 'correct', 'such', 'only', 'different', 'other', 'several', 'some', 'available', 'satisfactory', 'perfect', 'significant', 'valuable', 'harmful', 'unseen', 'good', 'multiple', 'appendix', 'number of', 'others', 'new', 'many', 'further', 'certain', 'the'