



National Research
Council Canada

Conseil national
de recherches Canada

Institute for
Information Technology

Institut de technologie
de l'information

NRC - CNRC

Learning the Ontological Positions of Natural Language Objects *

Wang, Y.
September 2003

* published as Master's Thesis, Presented at the Faculty of Computer Science,
University of New Brunswick, Fredericton, New Brunswick, Canada. September 18, 2003.
NRC 46523.

Copyright 2003 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report,
provided that the source of such material is fully acknowledged.

LEARNING THE ONTOLOGICAL POSITIONS OF NATURAL LANGUAGE OBJECTS

by

Yue Wang

BA — Beijing No.2 Foreign Language Institute, P. R. China, 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

in the Graduate Academic Unit of Computer Science

SUPERVISORS: Prof. Bruce Spencer, Faculty of Computer Science
Prof. Huajie Zhang, Faculty of Computer Science

EXAMINING BOARD: Dr. Yuhong Yan, Faculty of Computer Science
Prof. Lev Goldfarb, Faculty of Computer Science
Prof. Anna Maclachlan, Dept. of Culture and Language Studies

This thesis is accepted.

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

September, 2003

© Yue Wang, 2003

Abstract

This thesis endeavors to solve a text classification (TC) problem of a real-world system, New Brunswick Opportunities Network (NBON), an online tendering system that helps the vendors and the purchasing agents to provide and obtain information about business opportunities. The solution mainly involves techniques in the areas of machine learning and natural language processing (NLP). We use a Naïve Bayes classifier, a simple and effective machine learning approach for TC tasks, to automatically classify the tenders of the NBON system. We implement three smoothing algorithms for the Naïve Bayes classifier, namely, no-match, Laplace correction, Lidstone's law of succession, and we show that the difference between the accuracies obtained for the three algorithms is negligible. We show that the effectiveness of the Naïve Bayes classifier is better than that of three other TC techniques that are equally simple, namely, Strong Predictors (a modification of Term Frequency), TF-IDF (Term Frequency - Inverse Document Frequency), and WIDF (Weighted Inverse Document Frequency). NLP tools such as stop lists and stemmers are adopted for the text operations on the historic NBON data that is used to train the classifiers. We experiment with variations of such tools and show that NLP techniques do not have much impact on the effectiveness of a classifier.

Contents

Abstract	ii
List of Tables	vi
Acknowledgements	ix
1 Introduction	1
1.1 Ontologies	1
1.2 Text Classification	2
1.3 Machine Learning	3
1.4 Natural Language Processing	4
1.5 Thesis Rationale and Objective	5
2 The NBON System	6
2.1 How Does the Current NBON System Work?	6
2.1.1 The Structure and Patterns of GSIN Codes	9
2.1.2 Example Tenders of the NBON System	10
2.2 What Can Be Done to Improve the Usability of the Current NBON System?	12
2.2.1 Machine Learning	13
2.2.2 Natural Language Processing	14
2.2.3 Improvement on the NBON System	16

3	Naïve Bayes Classification with Natural Language Processing	18
3.1	The NBON Sample Data	18
3.2	Measure of Effectiveness and Evaluation Method	19
3.2.1	Measure of Effectiveness	19
3.2.2	Evaluation Method	20
3.3	NLP Techniques Used with the NBON Data	22
3.3.1	Feature Selection	22
3.3.2	Stemming	23
3.4	Text Classification using Naïve Bayes Technique	24
3.4.1	Bayes Theorem	24
3.4.2	Naïve Bayes	25
3.4.3	The Algorithm for the Naïve Bayes Classifier	26
3.5	Particular Problems with the NBON Data	29
3.5.1	Problem 1: Large Number of Categories	29
3.5.2	Problem 2: Insufficient Number of Tenders	30
3.5.3	Problem 3: Uncovered Categories	32
3.6	Design Choices	33
3.6.1	Hierarchical Classification with Top-level Categories	33
3.6.2	Classification with Categories Having Many Tenders	36
3.6.3	Classification with Ranking	38
4	Experiments	40
4.1	Experiments on the Naïve Bayes Classifier	40
4.1.1	Three Smoothing Algorithms	40
4.1.2	Effect of Data with Different Sizes	46
4.2	Non-Bayesian Text Classification Techniques	49
4.2.1	Strong Predictor Classifier: a Modified TF Classifier	50
4.2.2	TF-IDF Classifier	52

4.2.3	WIDF Classifier	54
4.3	Natural Language Processing Techniques	56
4.3.1	Feature Selection Methods	57
4.3.2	Stemming Algorithms	60
4.4	Comparative Results and Error Analysis	66
4.4.1	Comparative Results	66
4.4.2	Error Analysis	69
5	Conclusion	72
5.1	Machine Learning Techniques for Text Classification	72
5.2	Natural Language Processing Techniques for the Naïve Bayes Classifier	73
6	Future Work	75
6.1	Improvement on the Reliability of the Data	75
6.2	Recognition of French Tenders	75
6.3	Lower-level Hierarchical Classification	76
6.4	Classification of Vendors	77
6.5	Classification of Services Tenders	77

List of Tables

1.1	A Checkers Learning Problem	3
1.2	A Text Classification Learning Problem	4
2.1	A Long Tender	11
2.2	A Medium-sized Tender	12
2.3	A Short Tender	12
3.1	Accuracies Obtained for Each Round in a 20-fold Cross Validation . .	21
3.2	The Algorithm for a Naïve Bayes Classifier [Manning and Schütze 1999]	27
3.3	Characteristics Comparison of the Original and Top-level Categories .	34
3.4	Accuracy Comparison of Non-hierarchical and Top-level Hierarchical Classifications	36
3.5	Accuracy Comparison of Non-hierarchical Classification with Different Sample Data	37
3.6	Accuracy Comparison of Top-level Hierarchical Classification with Dif- ferent Sample Data	37
3.7	Accuracy Comparison of Top-level Hierarchical Classification with Rank- ing	39
4.1	Accuracy Comparison of Top-level Hierarchical Classification with and without Smoothing	41

4.2	No-match – Accuracy Comparison of Top-level Hierarchical Classification with Different f Values, best accuracies in bold	43
4.3	Laplace Correction – Accuracy Comparison of Top-level Hierarchical Classification with Different f Values, best accuracies in bold	44
4.4	Lidstone’s Law of Succession – Accuracy Comparison of Top-level Hierarchical Classification with Different f Values, best accuracies in bold	45
4.5	Characteristics of Three NBON Data Sets	47
4.6	Accuracy Comparison of Top-level Hierarchical Classification on Three Data Sets	48
4.7	Example Top-level Goods Categories and Their Unique Predictors . .	51
4.8	Accuracy Comparison of the SP and Naïve Bayes Classifiers for NBON Data	52
4.9	Term-Document Matrix for TF-IDF	52
4.10	Accuracy Comparison of the TF-IDF and Naïve Bayes Classifiers for NBON Data	54
4.11	A TF-IDF Example with a 5-Category Data Set	55
4.12	Accuracy Comparison of WIDF and Naïve Bayes Classifiers for NBON Data	56
4.13	Accuracy Comparison of Naïve Bayes Classifiers using Different Feature Selection Methods	57
4.14	Accuracy Comparison of SP Classifier using Different Feature Selection Methods	58
4.15	A 36-Word Stop List [Stop List]	59
4.16	Accuracy Comparison of Naïve Bayes Classifications using Different Stop Lists	59
4.17	An Example Error Made by a 36-word Stop List	60
4.18	Accuracy Comparison of Naïve Bayes Classifiers using Different Stemming Algorithms	64

4.19	A Misclassification Example of the Porter Stemmer	65
4.20	Accuracy Comparison of Naïve Bayes Classifiers With and Without Stemming	66
4.21	Accuracy Comparison of all the Text Classification Algorithms in this Thesis	67
4.22	Accuracy Comparison of Naïve Bayes Classifications on Three Different Data Sets I	67
4.23	Accuracy Comparison of Naïve Bayes Classifications on Three Different Data Sets II	68
4.24	Accuracy Comparison of Naïve Bayes Classifications using Different NLP Techniques	69

Acknowledgements

I would like to thank my supervisors, Professor Bruce Spencer, for all of his help and support during the production of this thesis, and Professor Huajie Zhang, especially for his technical directions in machine learning. Many thanks to Professor Anna Maclachlan for her insight and ideas in natural language processing, and Dr. Yuhong Yan for her valuable inputs in the NBON project. Also deserving thanks are all of my family and friends.

Chapter 1

Introduction

During the last decade, commercial interests in the research areas such as ontologies, text classification and natural language processing have been kindled and increased by the fast growth of the supply and demand of information on the Web. In the business world, timely acquisition of information is crucial, and the Internet has made such acquisition more convenient, but at the same time overwhelming. As a result, commercial sites and tools to help people obtain information that meets their interests have been getting more and more popular.

1.1 Ontologies

What are ontologies?

In recent years, ontology has become a popular word within the community of knowledge engineering, the research area that deals with the management and usage of knowledge. What is an ontology? The term “ontology” originated from philosophy and was used to refer to a systematic account of existence. In knowledge engineering, such existence is reflected in the representability of a domain by formal vocabulary [Gruber 1993]. A short definition of an ontology given in [Gruber 1993] is that it

is “an explicit specification of a conceptualization”. From a commercial viewpoint, people are more interested in what an ontology is used for.

What are ontologies used for?

Gruber and his colleagues defined the purpose of an ontology as “to enable knowledge sharing and reuse” [Gruber 1993], which is done through organization of the knowledge to maximize the efficiency of the retrieval. In this sense, ontology can be viewed as the taxonomy of a knowledge base. Two real-life applications of such taxonomies include the Yahoo! directory for information retrieval (IR) and the product catalogs of online shopping sites such as amazon.com.

It can be seen that the main issue of these applications is to organize the knowledge into a taxonomy, which is typically a tree-like hierarchical structure. Therefore this is a text classification problem, with the texts as the knowledge to be classified under predefined categories (i.e. domains) in the knowledge base.

1.2 Text Classification

Text classification (TC) is the task of sorting documents by certain criteria, e.g. content, author, publisher, language, etc. [Jackson and Moulinier 2002]. Both the examples mentioned in Section 1.1 involve sorting documents according to their content. This is given a special term, text categorization, by some researchers. In this thesis, we do not make a distinction between the terms text classification and text categorization, and use them interchangeably, although, strictly speaking, we actually only mean text categorization.

Like many other tasks, TC was traditionally performed by humans (called human indexers), who read the documents and assign predefined labels or index terms to them. Since manual TC requires great effort, is time-consuming and error-prone, intense academic attention has been paid since the last decade [Jackson and Moulinier 2002]

to write computer programs (called classifiers) to learn the categorization rules and automatically classify documents.

1.3 Machine Learning

Machine learning is the study of computational models and the supporting algorithms that can learn through experience [Mitchell 1997].

Although we can not yet make computers learn nearly as well as humans do, the way in which a computer learns is very similar to that of humans in that both learn through experience, i.e. they become more confident in positive experiences and learn lessons from negative experiences. Consider a computer program that learns to play checkers: by playing games against itself, a computer can gain experience that can be used to improve its skills. This is actually very similar to how a human player (of checkers or anything else) learns to improve his skills through practice.

A machine learning problem is usually defined by the class of tasks, the measure of performance, the source of experience, and the target function [Mitchell 1997]. Table 1.1 is the definition of a checker learning problem given in [Mitchell 1997].

A checkers learning problem:

- Task T : playing checkers
- Performance measure P : percent of games won against opponents
- Training experience E : playing practice games against itself
- Representation of the target function R : numbers and distributions of black and red pieces on the board

Table 1.1: A Checkers Learning Problem

In terms of TC, the most common approach is to employ *inductive learning* to make a computer program learn the categorization rules through a set of positive examples, i.e. documents labeled with correct categories. This is called *supervised*

learning, in contrast with *rote learning* where the categorization rules are simply given to the program, and *unsupervised learning* where categories are not predefined but have to be learned by the program, which will cluster the similar documents together [Jackson and Moulinier 2002].

It can be seen that the design issues of a machine learning problem typically involve choosing the training experience and the representation of the target function. Reflected in this thesis, the learning task can be defined as in Table 1.2.

A text classification learning problem:

- Task T : classifying texts under categories
- Performance measure P : percent of texts correctly categorized
- Training experience E : learning the representations of categories from texts labeled with correct categories
- Representation of the target function R : term-frequency pairs of all the words observed in each category

Table 1.2: A Text Classification Learning Problem

The details of the representation of the target function R will be discussed in Section 3.4.3.

Besides TC, machine learning also has wide applications in NLP, for example, parsing, part-of-speech tagging, etc.

1.4 Natural Language Processing

Natural language processing (NLP) refers to the computer-fulfilled functions used for the analysis (natural language understanding) or synthesis (natural language generation) of human language in spoken or written forms. The term “natural” is meant to contrast with formal languages (such as mathematical or logical notations) and computer languages (such as C and Java), which have been proven to be much more

comprehensible by computers [Jackson and Moulinier 2002]. In this thesis, we only address natural language understanding (NLU) in written forms, i.e. texts.

Understanding the meaning of a text, or the meaning of every word in the text, is challenging for a computer, if not for a human. Humans learn both the semantics and morphologies of words, whereas computers can only learn the morphologies easily, and when it comes to semantics, the problem becomes much more complex.

In this thesis, we do not address complex ambiguity in NLU, for example, the correct interpretation of metaphors and the situations that can just as easily confuse humans. For the purpose of TC, we only address briefly in Section 3.5.2 two common problems in NLU: synonymy and polysemy.

1.5 Thesis Rationale and Objective

In this thesis, the ideas of ontologies, machine learning and NLP are combined to solve the TC problems of a real-life system, New Brunswick Opportunities Network [NBON]. As will be discussed in Chapter 2, the main usability problem of the NBON system is the manual classification of users' documents, and we propose a machine learning approach to learn the categorization rules through historic data of the NBON system, which are the documents processed using some NLP techniques.

The structure of this thesis is as follows: Chapter 2 is an introduction of the NBON system, mainly about its current status, usability problems, and the potential solutions, which is the main topic of this thesis. Chapter 3 discusses a machine learning approach for text classification, the Naïve Bayes classifier. Chapter 4 discusses the experimentations carried out on the improved NBON system, including different smoothing algorithms for the Naïve Bayes classifier, the non-Bayesian classifiers, and the NLP techniques used to improve the effectiveness of the Naïve Bayes classifier. Finally, Chapters 5 and 6 are respectively the conclusion and the future work that may be performed.

Chapter 2

The NBON System

Doing business online is getting more and more popular these days. For example, online shopping is no longer a novel idea. The online environment where users browse the product catalog to look for particular products simulates a real life store where shoppers go through different sections to look for what they need. But looking for business partners on the Web, such as in an online tendering system, is a different story, because the navigation scope is much broader and users are often not as familiar with the structure of a tendering system as they may be with a product catalog, which is similar to the sections in a store.

This chapter introduces the NBON system, an online tendering system that helps the vendors and the purchasing agents find each other.

2.1 How Does the Current NBON System Work?

NBON provides the official online tendering system of the province of New Brunswick, which went live in May 2002. The motivation of the system was to help the vendors and the purchasing agents find each other online. The whole tendering and bidding procedure is similar to that of a traditional one, i.e. a purchasing agent submits a tender to the government (now represented by the NBON system), which notifies the

potential vendors, who can choose to bid, and finally, the purchasing agent chooses and announces the successful bidder. The whole process can be thus divided into four steps: registration, notification, bidding and announcement.

Registration

A vendor needs to electronically register with the NBON system to be recognised as a potential bidder. A purchasing agent submits a tender to the government, who puts it online. We call this the registration of the purchasing agent. Besides the traditional purpose of registration, another main target of this process is to classify the vendors and the tenders under certain categories, where each is uniquely represented by a GSIN code. Classification in the current NBON system is a manual process for both tenders and vendors.

Tenders. A purchasing agent submits to the government his request of a particular goods or service, called a tender. The tender is then assigned a GSIN code by a human indexer, either the purchasing agent or an expert in the government. If there are more than one item in the tender, each item will be treated as a separate request, and is given a separate code.

Vendors. A vendor needs to navigate the product catalog of the NBON system to pick up a GSIN code that properly describes his expertise. He may choose to classify himself on any level of the structure. Notice that the three super-level groups are not valid categories, because they do not have corresponding GSIN codes, and thus a vendor can not self-classify under, for example, goods. If a vendor is specialized in more than one field, he can self-classify under more than one category.

Notification

When a tender is submitted and manually classified, it will be published on the NBON system, and all the potential registered vendors of the goods or services in the tender will be identified by code matching and notified by electronic mail. The main

issue of notification is to retrieve the registered vendors with similar interests as the purchasing agent.

Since we do not want to leave out any potential vendor, the notification strategy of the NBON system is to notify all the registered vendors that share a path with the tender, meaning the parent, children and siblings of the category node under which the tender is classified. This strategy is made feasible by the hierarchical structure of the categories and their corresponding GSIN codes. The main reason behind such a strategy is that manual classification of the NBON system is not always reliable, therefore the system has to retrieve more vendors to be on the safe side. Recall that the vendors are self-classified, and since there are altogether 15,913 categories in the NBON product catalog, the vendors may be intimidated and take the shortcut to self-classify high in the hierarchical structure, and thus the classification result is less precise. The same thing happens to the human indexers, who, although more familiar with the categories, may not read the tender carefully or thoroughly enough to give each item the correct GSIN code, especially when such a human indexer is not the purchasing agent himself.

The obvious drawback of the strategy is that some vendors will occasionally receive false notifications. A superfluous notification of a tender that a vendor cannot supply is better than a missing notification of a tender that the vendor can supply, because the former can serve as a warning to the vendor that he may have classified himself under a wrong or too general category. And since the number of tenders per year is not very large in a province as small as New Brunswick, it is not a very serious issue if a vendor receives false notification occasionally. Besides, some vendors may desire more general notifications which may serve as free market information in the related areas.

Bidding and Announcement

The bidding and announcement procedures of the NBON system are the same as that of the standard offline ones, and are of no interest to this thesis, therefore will not be discussed in detail. Relevant information can be obtained on the website of the NBON system [NBON].

2.1.1 The Structure and Patterns of GSIN Codes

Similar to most product catalogs of online shopping sites, the NBON system has a hierarchical structure for its categories and their corresponding GSIN codes. The product catalog of the NBON system consists of three super-level groups: goods, services, and construction services. There are no GSIN codes applied to the three groups, and when we talk about the hierarchical structure, they are not included in, but actually on top of, the structure. Under the super-level groups, there are currently 15,913 categories grouped in up to four levels, each corresponding to a unique GSIN code. The total number of categories is not fixed but keeps on increasing, because new goods and services are being developed and thus new corresponding categories need to be added to the catalog. In terms of the structure of the product catalog, this means that the whole structure keeps on expanding and that the leaf nodes may become internal nodes. The following is a typical group of GSIN codes on the same path of the hierarchical structure and their corresponding categories:

N66 - Instruments and Laboratory Equipment
N6665 - Hazard-Detecting Instruments and Apparatus
N6665B - Detectors, Hazard
N6665BA - Detectors, Hazard, Concealed Weapon

Certain patterns of the GSIN codes can be recognised for the three super-level groups: goods codes either start with letter “N” followed by numerals, or start with “J” or “W” followed by another capital letter and then numerals; services codes

start with letters other than “N”, “J” or “W” followed by numerals; and construction services codes start with numerals followed by letters. Patterns also exist for the GSIN codes on each level of the category structure. Take goods categories for example, the GSIN codes on the first level (referred to as the “top-level categories” in this thesis) have two digits of numerals after the leading letter(s), and that on the second level have four digits, etc.

The hierarchical structure of the product catalog of the NBON system makes both registration and notification processes easier. For registration, the complexity of traversing a flat structure is N , with N denoting the total number of categories; such complexity is $\log(N)$ for a tree-like hierarchical structure. The notification process can be dynamically designed in a hierarchical structure, where the system can easily choose to notify the parent, children, and/or siblings of a node. The patterns of the codes on each level of each product group simplify the hierarchical traversal of the structure to pattern recognition, which means a computer program does not really need to construct a hierarchy of categories to benefit from such structure, but obtains the same convenience by working with GSIN codes of certain patterns.

2.1.2 Example Tenders of the NBON System

To illustrate the TC task of the NBON system, we show some example texts that need to be classified. In this thesis, we are only interested in classifying goods tenders, therefore we give three such tenders that were submitted to and published on the NBON system in Tables 2.1 through 2.3. The first row of each table shows the manually assigned GSIN code and the corresponding category of the tender.

In the NBON data of the two years from 1 January 2001 to 30 November 2002, which is the main data set used throughout this thesis, there are altogether 4,822 goods tenders classified under 1,269 categories. The biggest tender contains 826 words, and the smallest only 1 word. The average number of words per tender is 31.

Table 2.1 shows a rather long tender with 215 words. It gives detailed requirements

for the goods in question, such as functions, conditions, and measurements. Table 2.2 shows a medium-sized tender with 43 words. It also gives enough details about the request. The tender shown in Table 2.3 is very short, with 6 words only, and no detailed requirements are stated.

N6665 - Hazard-Detecting Instruments and Apparatus
<p>Corridor panels</p> <ul style="list-style-type: none"> - single corridor model of 2 panels - most operate on 110/120 VAC 60 Hz, Peak Power consumption 85 watts during alarm cycle less than 35 during normal operation - must be CUL approved - must reduce loss by 80% - to be directly mounted to the floor or installed with a base plate with buried cables - computerized electronic design to ensure accurate response to sensitized security strips - system must be able to protect : Print, CDs, audio cassette, video cassettes – system that is immune to electronic noise which can interfere with some systems - Safe for use, on all media, will not harm audio tapes, video tapes, computer diskettes - long term durability of the molded panels - audible, adjustable alarm and visible alarm triggered when materials removed without authorization - automatically conserves electrical power and eliminates unwanted alarms by searching for sensitized strips only upon exit of the patron - built in diagnostic code indicator for user friendly trouble shooting - 36" corridor to accommodate wheelchair patrons as stated by Americans with Disabilities Act (ADA) - multi-direction detection - must ensure that strips cannot be shielded by clothing, backpacks, etc. - sensor must be detected by photocells so that patrons cannot aid detection - Unit must be 44.5 inches W x 42 inches D X 67.5 inches H (3M Detection System model 2301 (Direct Mount) including installation)

Table 2.1: A Long Tender

N7110WAH - Filing Cabinet Insulated (Fire)
Lateral File with Lock, Fire Resistant Model GL404 Garde X Class "D" 1 hour - 1700 degrees F Description: -4 drawer -Height: 54" -Width: 37 3/4" -Depth: 22 3/4" -Weight: 850 lbs -Interior Dimensions of drawers: 32 1/2" W x 15 1/2" D x 10 1/2" H

Table 2.2: A Medium-sized Tender

JX66 - Instruments and Laboratory Equipment - Repair and Overhaul
Microscope, Compound, Boreal with Illuminator (55840-02)

Table 2.3: A Short Tender

2.2 What Can Be Done to Improve the Usability of the Current NBON System?

It can be concluded now that the main problem with the current NBON system is the manual classification of the tenders and the vendors, which both decreases the usability of the registration procedure of the system and increases the unreliability of notification.

Recall that in the current NBON system, the tenders are manually classified under one or more categories by human indexers. Classification is a tedious job for humans, and the quality of the work heavily depends on the moods of the human indexers, and thus are not always reliable. The situation is even worse for the vendors, because they need to self-classify under one or more categories without any help from either the system or the experienced human indexers. Some vendors are not familiar with the category structure of the NBON system, which means, in the worst case, they may have to go through all the 15,913 categories before identifying a proper one for themselves. Therefore, as we mentioned, some vendors chose to take a shortcut to

classify themselves high in the hierarchy.

Since manual classification is both time-consuming and error-prone, one popular approach of classification in recent years is inductive learning, i.e. to teach a computer the categorization rules and let it classify documents using the learned knowledge.

2.2.1 Machine Learning

Human versus Machine Learning

We mentioned in Chapter 1 that computers can not learn as well as humans. In the case of inductive learning, this is typically reflected in the learning of the rules and the induction of the conclusion. Humans know many things without being aware of the existence of such prior knowledge in our memories. Such prior knowledge typically includes some “common sense” facts and the knowledge in related areas that can be helpful in the induction process of the current learning task. But computers do not have any common sense at all, and they usually only learn one task at a time, and therefore they do not have any side knowledge to help them in induction.

However, computers have some advantages over humans. The original motivation for the invention of computers was to compute, and therefore computers are far more advanced than humans in computing speed, which means given the same knowledge that is necessary in solving a problem, computers are faster than humans in replying and reacting. Reflected in a TC task, such reactions typically refer to the efficient examination of a very large search space in order to find the optimal match. In addition, computers do not “forget” what they have learned and stored in their memories, whereas humans do. One successful example of machine learning is the famous TD-gammon [Tesauro 1995], the world’s top computer program for playing backgammon, which learned its strategy by playing over one million practice games against itself, and now plays at a level competitive with the human world champion. Such success encourages us to believe that although computers make mistakes that a human would

not make, they make up in the areas where humans tend to err. Therefore we can expect that computers perform nearly as well as humans for some tasks, while they are much cheaper and never complain of being bored and tired.

Machine Learning Techniques for TC Tasks

Many machine learning techniques have been developed for TC tasks. The most commonly used ones include, just to name a few, Naïve Bayes classifiers, decision trees, nearest neighbours algorithms, support vector machines and neural networks [Sebastiani 2002]. Each of these techniques has advocates in both academic and business worlds. Although comparisons of the effectiveness of these techniques have been carried out, no common agreement has been achieved as to which one is the best [Sebastiani 2002, Mitchell 1997, Jackson and Moulinier 2002].

We chose Naïve Bayes for the TC task in the NBON system, because it is the simplest algorithm among those mentioned above, and at the same time, its effectiveness was reported to be comparable to the more sophisticated techniques [Mitchell 1997].

2.2.2 Natural Language Processing

Unlike other machine learning tasks, TC learning is more complex because it involves understanding of natural language. The ambiguity in NLU results in two common problems: synonymy and polysemy. The solution to such problems is complicated and beyond the scope of this thesis. We address them in the Naïve Bayes algorithms and text operations using NLP techniques when applicable in our TC task.

Owing to the nature of the tenders, we do not expect to encounter complex NLU problems, such as metaphors, and thus simple text operations can be carried out for the purpose of classification. The common problems with any kind of text include stop words and word variations.

Stop Words and Content Words

A stop word is one that does not have a specific meaning. This typically includes all the function words, such as articles, propositions and conjunctions, and some frequently used words, such as “do”, “be”, etc. Content words, on the contrary, are the ones that have specific meanings, which typically include nouns, verbs, adjectives and adverbs.

When reading a text, a human knows that content words contribute more to the semantics of the text, and that stop words only play an auxiliary role. But a computer, which “learns” words by their morphologies, can not automatically tell the difference between a stop word and a content word in terms of their contribution to the semantics of the text. Owing to the fact that a computer sees the content of a text as being represented by words and their frequency, the frequently present stop words can easily mislead a computer program when reading a text.

Word Variations and Derivations

A human knows that the words such as “categories”, “categorization” and “categorized” are simply variations or derivations of the word “category”, and thus will make connections when reading a text containing such words. But a computer will not recognise that these morphologically different words are semantically similar unless they are processed to look the same. It can be seen that the failure of a computer in recognising words from the same root can result in a form of synonymy, i.e. words used in different forms to express similar or related ideas.

To teach a computer to “understand” a text in an intelligent way, e.g. ignoring stop words and identifying words of common root, some operations are usually carried out on the original text before feeding them into a computer program. Such text operations typically include pruning the stop words and stemming the content words. Text operations will be discussed in detail in Section 3.3.

2.2.3 Improvement on the NBON System

Automatically Classify Tenders

The motivation of this thesis is to make the classification process automatic, or at least, semi-automatic using a machine learning approach, i.e. to learn from the historic data of the NBON system and use the learned knowledge to classify a new document. We start with the classification of the tenders, because we can easily obtain the training data necessary for the learning approach that we are to take in this thesis: the tenders and their manually assigned categories from the past years are perfect training examples that we can feed to a computer program to build a classifier based on the categorization rules that it learns from the historic data.

In this thesis, only the goods tenders are selected as the sample data. This is because the code structures of the three super-level groups are different from one another. Working with only one group of examples makes it easier to perceive the code patterns on different levels. Goods is a good group to start with because goods data is more representative than the other two in terms of both overall quantity and number of levels. Taking the NBON data of the last two years as an example, the goods data covers more than 60% of the total tenders and nearly 80% of the total categories.

Create and Classify Vendors' Product Descriptions

Optimally, the system would automatically classify both the tenders and the vendors. Unfortunately, the current NBON system does not give the vendors a chance to describe their expertise, and thus there are no vendors' data available for the learning purpose. Therefore the user interface of the current NBON system will need to be redesigned to allow vendors to create and submit their product descriptions to be classified.

Before such product descriptions from the vendors can accumulate and be obtained

for the training purpose, our plan is to build the classification system based on the tenders data, and use this system to attempt to classify both tenders and vendors, assuming that the two use the same language when they are talking about the same thing. This assumption is safe, since when people do business, the buyer and the seller have to know what each other is talking about, and thus tend to use the same language. On the other hand, it would not be surprising if the vendor tends to use more professional jargons, whereas the purchaser, who may not be familiar with what he is going to buy, uses totally different terms. In order to solve this problem, the vendors may be asked to provide, instead of their product descriptions, the tenders of which they would like to be notified, provided that such tenders are available.

Redesign of the user interface is not the task of this thesis. Once this is done, the classification of vendors should be carefully tested, since our vendor classification assumption may turn out to be wrong. Nevertheless, the initial goal is not to classify the vendors with high accuracy, though it would be a bonus if it does, but to get vendors data so that, if necessary, a more reliable classifier can be built separately for the vendors.

Chapter 3

Naïve Bayes Classification with Natural Language Processing

3.1 The NBON Sample Data

Throughout this thesis, the data set used to train and test all the classification techniques are tenders and their corresponding categories manually assigned by the human indexers. The tenders and the categories have a many-to-one correspondence, which means each tender is labeled with one, and only one, GSIN code. In the case that a tender contains goods and/or services that actually belong to multiple categories, such a tender is divided into multiple tenders, so as to preserve the many-to-one correspondence between tenders and categories.

Throughout this chapter, the Naïve Bayes classifier is trained and tested on a data set consisting of the goods tenders published in the NBON system during the two years from 1 January, 2001 to 30 November, 2002.

3.2 Measure of Effectiveness and Evaluation Method

3.2.1 Measure of Effectiveness

Accuracy is the sole measure of effectiveness for all the classification and NLP techniques throughout this thesis. Accuracy tests how well a classifier can predict the category for a previously unseen tender.

$$Accuracy = \frac{\text{the number of tenders that are correctly classified}}{\text{total number of tenders}} \quad (3.1)$$

Recall and precision are two commonly used measures for IR tasks, which test how well a search engine can retrieve documents that are relevant to a user's query.

$$Recall = \frac{\text{the number of relevant documents retrieved}}{\text{total number of relevant documents}} \quad (3.2)$$

$$Precision = \frac{\text{the number of relevant documents retrieved}}{\text{total number of retrieved documents}} \quad (3.3)$$

Recall and precision were also used by some researchers [Fukumoto and Suzuki 2002, Tokunaga and Iwayama 1994, Yang and Pedersen] for TC tasks. In fact, IR can be viewed as a special case of TC, in which the texts are classified under either of the two classes, the ones that are relevant to a user's query and those that are not. It can be seen in Equations (3.1) and (3.3) that accuracy is equivalent to the standard precision measure under the one-category-per-document assumption. The definition of recall in TC is the ratio of correct assignments by the classifier divided by the total number of correct assignments [Fukumoto and Suzuki 2002], and therefore accuracy is also equivalent to recall in our case since we assume that the tenders used for testing are correct assignments.

Tokunaga and Iwayama defined recall and precision for their TC tasks in the following way [Tokunaga and Iwayama 1994]:

$$Recall = \frac{\text{the number of texts that are assigned to the correct category}}{\text{the number of total texts in the category}} \quad (3.4)$$

$$Precision = \frac{\text{the number of texts that are assigned to the correct category}}{\text{the number of total texts that are assigned to the category}} \quad (3.5)$$

The recall and precision values are calculated for each category as described in Equations (3.4) and (3.5), and the average over the whole collection is obtained as the final results.

There are two main reasons why we use accuracy as the sole measure to test the effectiveness of our classifier. First, in our case, what the purchasing agents want is to find a category that fits their tenders best, but not to get as many relevant choices as possible, which is typically desired in IR tasks. Second, to compare with the database of an IR system, for instance, the number of documents on the web or in a library, the total number of candidate categories, which can be viewed as the database for a TC system, is very small, and the number of categories on each hierarchical level is even smaller, therefore recall and precision defined in Equations (3.4) and (3.5) are not very useful measures for the effectiveness of our classifier.

3.2.2 Evaluation Method

The evaluation of the effectiveness of each technique implemented in this thesis is carried out with a 20-fold cross validation. The data set is divided into 20 equal folds, and the system runs 20 rounds. In each round, one fold is held out as the testing data and the rest serves as the source for the training data. The classifier is trained on the training data, and each tender in the testing data is used to test the effectiveness of the classifier. At the end of each round, an accuracy value is calculated according to Equation (3.1). Finally, the accuracy values obtained in all the 20 rounds are averaged to get a final accuracy result for the classifier.

In another commonly used evaluation method, train-and-test, a certain proportion of the available data is held out as the testing data, and the classifier is trained by the rest of the data. For instance, in [Joachims 1996] two-thirds of the sample data were

used to train the classifier, and the effectiveness of the classifier was measured over the remaining third. In [Blei et al 2003], the proportions of the training and testing data are 90% and 10% respectively.

The obvious advantage of the k -fold cross validation is that each tender in the data set will get a chance to serve as the source of testing, so that within the available data, an exhaustive test is carried out to evaluate the classifier. With the train-and-test approach, however, there can always be some uncommon cases in the data set that never get a chance to be tested, and therefore the result is not as reliable as the one obtained using k -fold cross validation. The unreliability of the train-and-test approach is verified by the accuracy results obtained in the 20 rounds of a 20-fold cross validation as shown in Table 3.1. Notice that the accuracies may be close to 100% in some cases, but such accuracies are not reliable, because the final accuracy averaged over the 20 rounds is only 91.34%.

20-fold round	accuracy	20-fold round	accuracy
Round 1	86.60%	Round 11	95.88%
Round 2	91.75%	Round 12	85.57%
Round 3	98.97%	Round 13	92.78%
Round 4	96.91%	Round 14	82.47%
Round 5	91.75%	Round 15	92.78%
Round 6	85.57%	Round 16	88.66%
Round 7	92.78%	Round 17	94.85%
Round 8	95.88%	Round 18	96.91%
Round 9	93.81%	Round 19	85.57%
Round 10	90.72%	Round 20	86.60%

Table 3.1: Accuracies Obtained for Each Round in a 20-fold Cross Validation

The accuracies shown here are obtained from the 2-year NBON data consisting of categories with more than 200 tenders each. The details of such experiments are introduced in Section 3.6.2.

3.3 NLP Techniques Used with the NBON Data

Text operations on a data set refer to processes such as part-of-speech tagging, stemming, etc. In this chapter, the NLP techniques used with the NBON data (both the training and the testing data) affect the decision rules on what to keep as the features for a category, called feature selection, and the stemming of the selected feature words.

3.3.1 Feature Selection

The feature selection method used in this chapter is the pruning of words of high frequency, called stop words, which typically include function words, such as articles, prepositions, pronouns, conjunctions, etc., and some commonly used words, such as “do”, “well”, “year”, etc. There are two main reasons why we want to eliminate stop words. One is to cut down the size of the feature space, the other is to prevent ubiquitous words from getting too much weight, since such words tend to have high *term frequency (TF)*, and Naïve Bayes classifiers, unlike TF-IDF classifiers, do not directly address *inverse document frequency (IDF)*. The details about Naïve Bayes and TF-IDF classifiers will be discussed respectively in Sections 3.4 and 4.2.2.

A part-of-speech (POS) tagger was originally used in this thesis for the purpose of removing the stop words. A POS tagger tags every word in a document with its POS feature according to its context, so that a feature selection method can choose which POS categories of words shall be removed from the feature space. There are three main disadvantages in using a POS tagger to select features. First, the stop words can only be recognised by their POS features, such as articles and prepositions, but the frequently used content words mentioned above can not be identified. Second, POS tagging itself usually involves some computations, and thus can make such a process rather expensive. It is not uncommon that a word has different POS features in different contexts, and therefore the tagging process itself may involve

some learning. For instance, Eric Brill's well-known POS tagger uses a learning approach to automatically generate a set of rules, and then uses such rules to tag the words in a new document with their POS features. This is obviously too elaborate a process for the purpose of pruning the stop words. Finally, due to the learning process and the computations involved in a POS tagger, the performance of the classification is inevitably worsened.

To compare with a POS tagger, a stop list, which is cheap, fast and easy to use, is preferable for the purpose of feature selection. In this chapter, a stop word list from the Defense Virtual Library [DVL/Verity] consisting of 456 words is used to prune the stop words in the NBON data. The selection of the 456 words was based on word frequency counts, and thus includes both function words and some frequently used content words.

In addition, owing to the nature of the NBON tenders, prevalent features such as dates, item numbers, quantities, measurements, etc., are also removed from the data set, because they usually do not tell anything about a particular category, and thus are not good features.

3.3.2 Stemming

Stemming is a commonly used process in IR. The original motivation of stemming is to reduce the impact of variations of the same word on the word matching process, which is essential in classification. Since variations of the same word are reduced to their stemmed form, the feature space will be further cut down.

Once the features are selected using the feature selection method previously mentioned, a stemming algorithm is used to remove the suffixes from the content words, such as the suffixes for the conjugated verbs, the plural forms of nouns, the comparative and superlative forms of adjectives, the derivations and variations of the same word, etc. There are two frequently used stemming algorithms in NLP: Lovins stemmer [Lovins 1968] and Porter stemmer [Porter 1980]. The Lovins stemmer removes

more than 290 suffixes using a longest-match algorithm, and the Porter stemmer removes about 60 suffixes using a multi-step approach. Although the great difference between the numbers of suffixes removed by the two stemming algorithms mainly owes to the combinatorical effect of the suffixes in the Lovins stemmer, the Porter stemmer, which was developed later than the other, had obviously realized some disadvantages in removing too many suffixes, and therefore shrank the number of removable suffixes. For example, suffixes such as “ish”, “hood”, etc., are not removed by the Porter stemmer. The advantages and disadvantages of the two stemmers will be discussed in detail in Section 4.3.2.

In this thesis, the Porter stemmer is used to stem words. The Porter stemmer uses a simple set of rules to reduce a word to its stem in multiple steps. For example, the word “meetings” would first be reduced to “meeting” and further to “meet”.

3.4 Text Classification using Naïve Bayes Technique

3.4.1 Bayes Theorem

Machine learning tasks typically involve determining the best hypothesis h_{best} from a hypothesis space H , given the observed data D . A straightforward way to define the *best* is through its probability, i.e. the *most probable* hypothesis h in H , given the data D plus any initial knowledge about h . Such probability of h is called the *posterior probability* of h , denoted by $P(h|D)$. The initial knowledge about h is referred to as the *prior probability* of h , denoted by $P(h)$. And $P(D)$ denotes the probability of the observed data D , $P(D|h)$ denotes the probability of observing D given some world in which h holds. [Mitchell 1997]

Bayes theorem provides a direct method to calculate $P(h|D)$, based on $P(h)$,

$P(D|h)$, and $P(D)$:

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)}. \quad (3.6)$$

Since $P(D)$ is a constant independent of h , it is usually dropped off, and the equation is thus simplified to

$$h_{best} = \max_{h \in H} P(D|h) \cdot P(h), \quad (3.7)$$

when looking for the best hypothesis in H . If we assume all the individual data points d_i in D to be mutually independent, the term $P(D|h)$ can be written as the product of all $P(d_i|h)$ terms, and thus Equation (3.7) becomes

$$h_{best} = \max_{h \in H} P(h) \cdot \prod_{d_i \in D} P(d_i|h). \quad (3.8)$$

Equation (3.8) is the decision rule applied throughout this chapter.

3.4.2 Naïve Bayes

Naïve Bayes is the simplest method in Bayesian learning, a well recognized family of machine learning techniques based on Bayes rule. Despite its simplicity, Naïve Bayes learning has been proven to be a very effective learning method, and thus is widely used for practical problems. In some domains, its effectiveness has been shown to be comparable to that of neural network and decision tree learning, which are far more complex in terms of algorithms [Mitchell 1997].

In machine learning tasks, d_i is usually represented by a set of feature-value pairs. Since D typically consists of multiple d_i individuals, D is in turn represented by a set of feature-value pairs, because Naïve Bayes assumes that the order of the d_i individuals is not important, and all the features are mutually independent (and thus the name “naïve”). A Naïve Bayes classifier learns the conditional probability of each feature f_i given the hypothesis h in the training data, and then applies Bayes rule to compute the posterior probability of h given the particular instantiations of the

features in the testing data. Therefore, the $P(D|h)$ term in Equation (3.7) can be conveniently computed as the product of all $P(f_i|h)$ terms.

Now, let us go back to our decision rule, Equation (3.8), and discuss how to get the terms required in the equation. The prior probability $P(h)$ is easy to compute as the ratio of the frequency of h , denoted by $F(h)$, in the data set divided by the data size.

$$P(h) = \frac{F(h)}{|D|} \quad (3.9)$$

The computation of the conditional probabilities $P(f_i|h)$ is more complex. By the product rule, we have

$$P(f_i \wedge h) = P(f_i|h)P(h), \quad (3.10)$$

therefore we can get $P(f_i|h)$ by rewriting Equation (3.10) as follows

$$\begin{aligned} P(f_i|h) &= \frac{P(f_i \wedge h)}{P(h)} \\ &= \frac{F(f_i \wedge h)/|D|}{F(h)/|D|} \\ &= \frac{F(f_i \wedge h)}{F(h)}. \end{aligned} \quad (3.11)$$

Now we have all the terms required in Equation (3.8) where $P(d_i|h)$ is replaced by $P(f_i|h)$, and all the terms can be conveniently obtained by counting frequencies.

3.4.3 The Algorithm for the Naïve Bayes Classifier

Text classification, or text categorization, is different from other machine learning problems in that there is no obvious feature-value representation in the training examples from which a learner can learn about a concept, which is a category in the case of TC. Naïve Bayes method makes such representation “disturbingly simple” [Mitchell 1997] by defining each distinct word to be a feature, and the frequency of the word in a document to be its value. A category is represented by all the documents (i.e. tenders in our case) in a data set that are labeled by such category.

This is called “a bag of documents”, which implies that the order of the documents is irrelevant. Since the order of the words in the documents is also ignored, a category is thus represented in turn by a bag of words.

```

1   % the training phase
2   % input: tender  $t$  labeled with category  $c$ 
3   % output:  $P(w|c)$  for all words  $w \in t$ ,  $P(c)$  for all categories  $c$ 
4   for all categories  $c_k$  do
5       for all words  $w_j$  in the set of tenders  $t_{c_k}$  labeled with  $c_k$  do
6            $P(w_j|c_k) = F(w_j, c_k)/F(c_k)$ 
7       end
8   end
9   for all categories  $c_k$  do
10       $P(c_k) = F(c_k)/\sum_{i=1}^n F(c_i)$ 
11  end
12  % the categorization phase
13  % input: a tender  $t'$ 
14  % output: the category  $c'$  predicted for  $t'$ 
15  for all the categories  $c_k$  do
16       $score(c_k) = P(c_k)$ 
17      for all words  $w' \in t'$  do
18           $score(c_k) = score(c_k) \cdot P(w'|c_k)$ 
19      end
20  end
21  choose  $c' = \max_{c_k} score(c_k)$ 

```

Table 3.2: The Algorithm for a Naïve Bayes Classifier [Manning and Schütze 1999]

Notice that the independence assumption that a Naïve Bayes classifier makes is obviously wrong for natural language, because words in a text are usually not totally independent of one another. This is typically true with noun and verb phrases, for instance in the noun phrase “natural language”, the two words are in a modification relation. In spite of such a false independence assumption, Naïve Bayes classifiers have been proven to be surprisingly effective, and it was reported in some studies that the accuracy does not always increase when word dependence is taken into account [Domingos 1996]. For instance, a semi-naïve Bayesian classifier was developed in

[Kononenko 1991] that attempts to join pairs of features based on statistical tests for independence, and the experimentation results were very disappointing.

Now let us look at the algorithm for a Naïve Bayes classifier. Since the tenders are natural language descriptions, which are ambiguous in many cases, the system needs to correctly interpret a description before classifying it correctly. In this sense, a TC task can be viewed as a word sense disambiguation (WSD) problem, i.e. a word is classified under different categories depending on its context. Table 3.2 is a modification of the Naïve Bayes algorithm for WSD in [Manning and Schütze 1999] adapted to our TC purpose.

As mentioned in Section 3.2.2, the 2-year NBON data is divided into 20 equal folds, and the above algorithm is run in 20 rounds to get a final accuracy result. Each round consists of the training phase carried out on the 19-fold training data, and the testing phase on the 1-fold testing data.

In the training phase (steps 4-11), all the tenders labeled with the same category are bundled as “a bag of documents” in the category, and all the words and their TF values are identified as the feature-value pairs for the category. Since the tenders and categories have a many-to-one correspondence, the frequency of each category is given by the number of tenders labeled with such a category. $P(c)$ and $P(w|c)$ are computed for all the categories and all the words in the training data.

In the categorization phase (steps 15-21), each tender t' in the testing data is used to test the Naïve Bayes classifier, and the category predicted by the classifier using the decision rule in Equation (3.8) is compared to the label of t' , the category that was originally assigned to t' by human indexers. An accuracy value is calculated when all the tenders in the testing data are tested, using Equation (3.1).

A final accuracy result is calculated by averaging over the 20 accuracy values obtained in the 20 rounds, which is reported as the accuracy of the classifier.

3.5 Particular Problems with the NBON Data

Naïve Bayes is known as an effective technique for TC tasks. For example, take some research works previously mentioned: the accuracy was reported to be 67% for [Tokunaga and Iwayama 1994] in Section 3.2.1 on, and 89% for [Joachims 1996] in Section 3.2.2 on page 21. However, applied to the NBON data that are processed by the feature selection and stemming operations mentioned in Section 3.3, the raw accuracy of the Naïve Bayes classifier is only 32.90%¹, which is definitely not acceptable for this application. It is not surprising though that the same technique can produce different results on different application problems. So does this mean that Naïve Bayes is not the proper technique for this particular application problem? To answer such a question, it is necessary to find out what contributed to the poor effectiveness of the Naïve Bayes classifier for the NBON system.

After analysing the NBON data, three data sparseness problems were identified, namely, large number of categories, insufficient number of tenders, and uncovered categories.

3.5.1 Problem 1: Large Number of Categories

In the 2-year NBON data set, there are altogether 1,269 categories, which is large for TC tasks. In the literature, the number of categories typically ranges from ten to one hundred. For instance, in [Joachims 1996] there are 20 categories, and in [Tokunaga and Iwayama 1994], the number of categories is 139.

It was observed in [Tokunaga and Iwayama 1994] that the accuracy increases with the decrease of the number of categories. The large number of categories is definitely the main contributor to the poor effectiveness of the classifier. Given 20 predefined categories, we would expect a random guess to achieve a classification accuracy of

¹All the accuracies reported in Chapter 3 are the results of using the no-match smoothing method, which, together with other smoothing methods, will be discussed in detail in Section 4.1.1.

5%, whereas with 1,000 categories, such random accuracy would be only 0.1%. In this sense, the 32.90% accuracy that we obtained is actually not bad, although it is definitely not reliable for the use.

With the Naïve Bayes algorithm, it can be easily imagined that when there are many features in the training set, the difference between two categories tends to be attenuated, because their $score(c)$ values tend to be close to each other. The more categories there are, the greater the chance that two categories get scores that are only insignificantly different from each other, and therefore when such score values are chopped to a certain precision, the category that originally had the highest score may not beat others, and thus will not be chosen by the classifier.

3.5.2 Problem 2: Insufficient Number of Tenders

A condition for a classifier to be reliable is that there be sufficient examples available for training. In the literature, the data set chosen for the evaluation of a classifier typically consists of many examples for each category [Mitchell 1997]. For instance, in the examples mentioned in Section 3.5.1, [Joachims 1996] had 1,000 examples for each of the 20 categories; and in [Tokunaga and Iwayama 1994], there were at least 20 examples in each of the 139 categories.

The importance of having many examples for each category is straightforward. It is intuitive that the more one learns about a concept, the better one knows about it. This is also true with machine learning. The more examples (i.e. tenders in our case) a classifier learns for a category, the better it can predict that category for a new tender. This was verified in some studies that have shown that accuracy increases with the number of training examples [Joachims 1996].

More examples can help reduce the influence of synonymy and polysemy, the two NLP problems that arise in many application areas, such as WSD, TC, IR, etc.

Synonymy refers to the fact that one concept can be expressed in multiple ways.

It is common that different people with different educational and professional backgrounds tend to use different words to address to the same concept. It may not always be difficult for a human being to understand a concept expressed in different ways, but it will be impossible for a machine to recognise a concept if the wording used is unknown to it. For instance, if a machine has never seen the word “categorization”, it will not be able to tell that it addresses the same concept that is expressed by “classification” in a given context, assuming that the machine knows the latter. This is actually also true for humans, who can not “recognise” something if they were not “cognizant” of it before. Machine learning does not solve the problem of synonymy because a machine only knows what it has learned. But the more a machine learns about a concept, the more possible wordings for such concept become familiar to it, and thus the chance is smaller that it encounters an unfamiliar expression about the same concept.

A sister problem of synonymy is polysemy, which refers to the situation that a word has multiple distinct meanings. A human being can correctly interpret a multi-meaning word through its context. The following two sentences are typical examples:

- (1) *He is walking on the bank of the Saint John River.*
- (2) *He is walking to the bank to deposit his pay cheque.*

A human being can easily tell that the “bank” in sentence (1) refers to the ground near a river, and that in sentence (2) refers to a financial establishment. So can a machine with the appropriate training. Imagine a machine learning system that is trained on the following two concepts represented by multiple features:

- (i) *the ground near a river: bank, river, etc.*
- (ii) *a financial establishment: bank, cheque, deposit, etc.*

The system should have no difficulties to categorize sentence (1) under category (i) and sentence (2) under category (ii). The Naïve Bayes technique provides a large context window for each concept by combining the evidence of all the features (i.e.

words and their frequencies) associated with the concept. It is obvious that the more examples a category has, the larger the context window will be, and thus the more it can tell about the category. Again, Naïve Bayes or any machine learning technique does not directly solve the problem of polysemy, but it can make the problem less serious.

Synonymy is closely related to the recall measure, since different wordings of the same concept may prevent some relevant categories from being retrieved. Polysemy has an important influence on the precision measure, because multiple meanings of the same wording may result in some noise being retrieved. Therefore it is natural that the accuracy, which has been shown in Section 3.2.2 as being equivalent to the standard recall and precision measures, is heavily dependent on the sufficient number of examples.

3.5.3 Problem 3: Uncovered Categories

Out of all the 14,783 goods categories corresponding to the GSIN codes for the NBON system, only 1,269 are covered by the 2-year data, which means the coverage of the sample data in terms of categories is only 8.58%. This will be a problem when the system under development is used in real life, because there is a great chance that it will encounter some categories that are not covered by the knowledge of the classifier.

In the development phase, this is not a problem, since we are working with the 1,269 goods categories only, for both training and testing. But among the 1,269 categories, 552 contain only one tender. These categories may not be covered in the training data when those particular tenders are held out as the testing data. In addition, there are another 432 categories that contain only 2 or 3 tenders, and since the fold size in our case is $\frac{4,822}{20} \approx 241$, there is a chance that those tenders fall into the testing fold if they happen to be close to one another. This would leave no tender in the training set, and thus would increase the number of the uncovered categories. In this sense, even in the development phase, the coverage of the categories in the

sample data can be rather low.

3.6 Design Choices

3.6.1 Hierarchical Classification with Top-level Categories

Recall that the categories and their corresponding GSIN codes are organized in a hierarchical structure. What we have done up to now has actually ignored such structuring characteristics of the NBON system and have treated the structure of the category space as a flat one. What if we classify a tender in a hierarchical manner, i.e. level by level?

There are three advantages of such a strategy that respectively address the three problems discussed in Section 3.5.

First, it dramatically cuts down the number of categories.

Recall that there are as many as 1,269 original goods categories in the sample data. There are, however, only 70 top-level goods categories, and fewer in each lower category level. This cuts down the number of categories dramatically. As we have discussed in Section 3.5.1, a smaller number of categories should help improve the accuracy.

Second, it increases the number of tenders in each category.

By combining the tenders in all its children nodes, a top-level category has more tenders to represent it. For instance, the proportion of the original categories with more than 10 tenders is only $\frac{70}{1,269} \approx 5.52\%$, such proportion in top-level categories, however, is $\frac{56}{70} = 80\%$. A comparison of the characteristics of the original and top-level categories can be found in Table 3.3. We use $|Categories|$ to denote the number of categories in the training data.

	original categories	top-level categories
$ Categories $	1,269	70
$F(c) \geq 10$	70	56
$F(c) \geq 50$	8	23
$F(c) \geq 100$	4	13
$F(c) \geq 200$	1	4
$F(c) < 10$	1199	14
$F(c) \leq 3$	984	7
$F(c) = 1$	552	4

Table 3.3: Characteristics Comparison of the Original and Top-level Categories

Increasing the number of tenders in each category also means at the same time decreasing the number of “sparse” categories, which are the ones shown in the last three rows in Table 3.3. For instance, there are 984 out of the total 1,269 original goods categories that have less than 3 (inclusive) tenders, with a proportion of 77.54%, and this number is cut down to 7 of the 70 top-level categories, which is only 10%.

Third, it decreases the number of uncovered categories.

Recall that the coverage of the original categories in the sample data is only $\frac{1,269}{14,783} \approx 8.58\%$. However, out of the total 72 top-level goods categories for the NBON system, there are only 2 categories that are not covered in the sample data, resulting in a coverage of $\frac{70}{72} \approx 97.22\%$. The two uncovered categories are

- (22) *Railway Equipment*
- (88) *Live Animals*,

where the numerals in brackets are the top-level GSIN codes for their categories.

In addition, there are 4 categories that have only one tender, which, as discussed in Section 3.5.3, may add to the number of uncovered categories. These four categories are

- (31) *Bearings*
- (60) *Fiber Optics Materials, Components, Assemblies and Accessories*

- (94) *Nonmetallic Crude Materials*
- (96) *Ores, Minerals and Their Primary Products.*

Although the classifier may not be able to correctly classify a new tender if it belongs to these six categories, the issue is actually not very serious, since these categories are rarely requested, which means the chance that the classifier encounters a tender that should be classified under such uncovered categories is small. Notice that except for category (88), all the other five categories listed here are goods that people do not request frequently. This supposition is also verified by observing more NBON data. For instance, only one more tender was added respectively to the categories (31) and (60) in the 4-year data (from 1 January 1999 to 30 November 2002), and both (22) and (94) were each requested only once in the last eight years. The details about the 4-year and 8-year NBON data will be discussed in Chapter 4.

Now that we know the advantage of hierarchical classification, is such a strategy feasible? The answer is “yes”. The hierarchical structure of the categories makes it convenient to classify the sample requests one level after another by observing the patterns of the GSIN codes. For instance, the first level of goods GSIN codes consist of 2 digits, and the second level 4 digits, etc.

Since it makes no sense to blindly go further down the category structure before we can confidently classify a tender on the higher levels, let us start with the hierarchical classification on the top-level categories.

It is shown in Table 3.4 that by working with the top-level categories, we can greatly increase the accuracy.

Although having greatly increased, an accuracy of 61.31% is by no means good, therefore it will not make sense to go down to the second category level before we can improve the accuracy on the top-level classification. Although top-level categorization is less fine-grained, we determined for our task that it is sufficient if we stop on the top level for now, and do not go further down the category structure at all. Recall that a

	all categories - flat	top-level categories only
$ Categories $	1,269	70
<i>Accuracy</i>	32.90%	61.54%

Table 3.4: Accuracy Comparison of Non-hierarchical and Top-level Hierarchical Classifications

tender notifies all the vendors classified on the same path, therefore no vendors will be left out when the tender is classified on the top level; only that some vendors will be unnecessarily notified. Getting irrelevant notifications is on one hand annoying, but on the other hand, it can be desirable in some cases, for instance, when the vendors change their fields of expertise before reclassifying themselves, and/or when they want to get some market information.

3.6.2 Classification with Categories Having Many Tenders

It has been shown in Section 3.5.2 that sufficient tenders in each category are important for a classifier to be reliable. Although we do get more tenders in each category by working with a hierarchical category structure, thus reducing the number of sparse categories, there are still categories that are unfamiliar, or even unknown, to our classifier. In this section, we are interested in how much the accuracy can be increased when we have more tenders for each category. Table 3.5 and 3.6 show examples of such increases. Here we use $|Tenders|$ to denote the total number of tenders in the training data. Looking back at Table 3.3, we can observe that in the original categories, there is only one category that has more than 200 tenders, and thus in Table 3.5, the experiments stop at the level of 100 tenders per category.

It can be observed that more tenders per category can make a great difference in terms of accuracy. For instance, both the last columns in both Table 3.5 and Table 3.6 represent data sets with $|Categories| = 4$, but the accuracy for the latter is 10.02% higher than that for the former, because it has more tenders in each category.

minimum $F(c)$	1	10	50	100
$ Tenders $	4,822	2,107	1,006	697
$ Categories $	1,269	70	8	4
<i>Accuracy</i>	32.90%	49.24%	76.70%	81.32%

Table 3.5: Accuracy Comparison of Non-hierarchical Classification with Different Sample Data

minimum $F(c)$	1	10	50	100	200
$ Tenders $	4,822	4,763	3,743	3,123	1,945
$ Categories $	70	56	23	13	4
<i>Accuracy</i>	61.54%	62.00%	72.78%	79.65%	91.34%

Table 3.6: Accuracy Comparison of Top-level Hierarchical Classification with Different Sample Data

Notice a counterexample, actually a bonus, in column 2 of Table 3.5 and column 1 of Table 3.6, where $|Categories| = 70$ in both cases, but the accuracy of the latter is 12.30% higher than that of the former, not the other way around as we would expect, based on the fact that the former has more tenders per category. The reason for this seemingly conflicted comparison result owes to the distribution of the tenders among the categories. Also notice that in Table 3.6, the differences in the numbers of tenders and categories between column 1 and 2 are small, whereas in Table 3.5, such differences are large. Therefore, in comparison with the original flat classification of all the categories, the effect of the sparse categories in top-level hierarchical classification is not very great. This can be verified by observing the accuracy differences between columns 1 and 2 in both tables: in Table 3.5, the difference is 16.34%, but in Table 3.6, it is only 0.46%.

By observing Table 3.3, we can see that the number of very sparse top-level categories, i.e. less than 3 (inclusive) tenders per category, is only 7, half of the total 14 categories that have less than 10 tenders, whereas the proportion for the

original categories is $\frac{984}{1,269-70} \approx 82.07\%$. This shows that by working with top-level categories, the accuracy achieved over all the categories can be even better than that for categories with more tenders in the original categories.

Now, by working with top-level categories that have many tenders, we have obtained accuracies that are roughly comparable to that of the two examples mentioned at the beginning of Section 3.5.

3.6.3 Classification with Ranking

Although an accuracy of over 90% can be achieved by working with top-level categories with more than 200 tenders, the result will not be very helpful in real life, because there are only 4 categories in the sample data, whereas the users' interests cover 72 categories (or 70 in the development phase).

Recall that up to now, we have only calculated the accuracy values in a very strict way, i.e. the classifier only returns the *best probable* category, the one with the highest score, and if such a category does not match the manually assigned one, it is wrong, and no second chance is given. Owing to the nature of Naïve Bayes classifier, especially that of the NBON data, the differences between any two $score(c)$ may not be very great, and thus the correct category may not get the highest score, but may rank 2nd or 3rd. The idea proposed here is to borrow the idea of ranking in IR to return to the user the top N categories ranked by scores, and let the user pick up the correct one.

Table 3.7 shows the accuracy increase that can be obtained by returning the N best categories to the users, instead of just the best category. It can be seen that at the level of top 10, the accuracy in each column is increased by approximately 20 percentage points. Column 5 has only two rows because there are only 4 categories in the training data, making it impossible to return the top 5 categories.

In the literature, only a few TC tasks (such as [Woods et al. 2000]) have measured the accuracy of the 10 best hits. Most reported accuracy results are those for the 1

minimum $F(c)$	1	10	50	100	200
$ Tenders $	4822	4,763	3,743	3,123	1,945
$ Categories $	70	56	23	13	4
1 Best Accuracy	61.54%	62.00%	72.78%	79.65%	91.34%
3 Best Accuracy	72.39%	73.28%	84.41%	90.58%	98.76%
5 Best Accuracy	76.43%	77.69%	88.48%	94.33%	
10 Best Accuracy	82.03%	83.63%	93.85%	98.27%	

Table 3.7: Accuracy Comparison of Top-level Hierarchical Classification with Ranking

best category. In these reports, an accuracy of, for instance, 90% is usually considered to be good. This may be true in research, but not in real world applications, because it means that the error rate is as high as 10%, which will disappoint the users and thus result in the poor usability of the system. It certainly would be preferable to have a completely automatic classification procedure, but when the 1 best prediction of a classifier is not very reliable, such automation is fairly meaningless. In such a case, it pays off to put the rights of the final decision to the user. Most users are accustomed to the ranking system of IR systems and would not mind having to select a relevant category among a list of ranked categories. Besides, this will give users a chance to classify their documents under multiple categories, which is desirable in some cases, such as when they have various requests in one tender, and/or when they want to know about other potentially relevant categorizations for their tenders. A major advantage of involving users in the classification procedure is to increase the accuracies for the top-level hierarchical classification and thus allow the finer grained classification in future phases.

Notice that our experiments stop at the level of top 10. This is because some studies of searching behaviour have shown that users tend to give up if a desired category is not found in the first ten ranked categories [Woods et al. 2000].

Chapter 4

Experiments

4.1 Experiments on the Naïve Bayes Classifier

4.1.1 Three Smoothing Algorithms

We have mentioned that the accuracies we obtained in Chapter 3 are for the Naïve Bayes classifier using no-match smoothing method. What does a smoothing method do? And what will happen if no smoothing method is adopted by a Naïve Bayes classifier?

In the testing phase, when a word w_j in a tender is not observed in the training phase of a category c_k , $TF(w_j, c_k)$ is zero, and thus the value of $P(w_j|c_k)$ zero. Since $score(c_k)$ is the product of all $P(w|c_k)$ terms, a single zero count will result in a zero score for c_k . The domination of a single non-observed word over the whole category is obviously improper.

In the TC literature, two kinds of non-observation are dealt with in different ways. One is unknowns, meaning that the word w_j was not observed in any category, the other is zero counts, meaning that w_j was not observed in a particular category c_k , but was observed elsewhere. In the case of unknowns, a normal approach is to simply ignore w_j , because it was actually not recognised as a feature in the training data,

and thus no machine learning method can do anything about it (since it was not “learned”). Zero counts are more complex. It is obviously improper to simply ignore w_j , since it was a recognised feature. Therefore, some smoothing methods have been developed in the literature to deal with zero counts.

Table 4.1 shows the sharp difference between the accuracies obtained for the Naïve Bayes classifier without any smoothing, and that for one using the no-match method. One may be surprised by the low accuracies obtained by the Naïve Bayes classifier without smoothing, especially those for the data sets consisting of categories with many tenders, because in such cases, the classifier should have learned enough about the categories and thus we should not expect the problem of zero-counts to be too serious. However, an examination of the NBON data set, for example, where minimum $F(c) = 200$, has shown that among the total 1,945 tenders, tested over 20 rounds, there are only 31 that do not have a zero-count in any of the four categories. This characteristic of the NBON data makes smoothing an important task of this thesis.

minimum $F(c)$	1	10	50	100	200
$ Tenders $	4,822	4,763	3,743	3,123	1,945
$ Categories $	70	56	23	13	4
Naïve Bayes without smoothing	22.50%	22.57%	28.26%	30.77%	34.59%
Naïve Bayes with no-match smoothing	61.54%	62.00%	72.78%	79.65%	91.34%

Table 4.1: Accuracy Comparison of Top-level Hierarchical Classification with and without Smoothing

In this section, we discuss three smoothing algorithms that deal with zero counts of recognised features. Each algorithm has advocates in the literature. We show here that, applied to the NBON data, the difference between the accuracies obtained for the three algorithms is trivial. All the three algorithms are implemented and tested with the same data set as in Chapter 3, and their 1 best accuracies of the top-level categorization are compared.

Algorithm 1: No-match

No-match is the simplest method that is used in many machine learning tasks to deal with zero counts. It simply replaces a zero count with a small number, for example, $0.5/F(c_k)$ in [Kohavi et al 1996]. Equation (4.1) is a general no-match method, in which 0.5 is replaced by f , a variable that can be adjusted under different circumstances to achieve the best accuracy result.

$$P(w_j|c_k) = \begin{cases} \frac{F(w_j,c_k)}{F(c_k)} & \text{if } w_j \text{ was observed in } c_k \\ \frac{f}{F(c_k)} & \text{otherwise} \end{cases} \quad (4.1)$$

Besides the original motivation of preventing $P(w_j|c_k)$ from being zero, f is at the same time a penalty factor that, when assigned a small number, can serve as a penalty for non-observed words that makes the probability value small.

In [Kohavi et al 1997], the optimal value of f was reported to be 0.01. Such a value actually varies for different application data sets. In the NBON data set, different subsets of the data are actually different application cases. For instance, an NBON subset in which each category has more than 10 tenders is different in both $|Categories|$ and $|Tenders|$ from a subset consisting of categories with more than 100 tenders each. Therefore the optimal f values are different for the different NBON subsets.

Table 4.2 shows the accuracy comparison of the no-match method tested on different NBON subsets under different values of f . The best accuracy value in each subset is emphasized in bold.

It can be observed that, as we would expect, in general, the optimal f value increases with the number of tenders per category. This is because when the $F(c_k)$ value is great enough (and thus the $\frac{1}{F(c_k)}$ value small enough) to be a penalty by itself, we do not need a very small f value to make it small enough to penalize the non-observed words in c_k . Unnecessarily heavy penalties will make things worse. This is verified by the decreases of the accuracies in Table 4.2 when f is too small.

minimum $F(c)$	1	10	50	100	200
$f = 0.1$	18.20%	49.31%	69.12%	78.72%	91.19%
$f = 0.01$	52.49%	60.34%	72.78%	79.65%	91.34%
$f = 0.001$	60.95%	61.95%	72.70%	78.85%	90.98%
$f = 0.0001$	61.54%	62.00%	72.49%	78.30%	90.57%
$f = 0.00001$	61.31%	61.74%	71.90%	78.08%	90.41%

Table 4.2: No-match – Accuracy Comparison of Top-level Hierarchical Classification with Different f Values, best accuracies in bold

Algorithm 2: Laplace correction

Laplace correction is a smoothing method widely used in TC tasks using Naïve Bayes. As far as we know, there are two different versions of Laplace correction in the literature.

Version I. The first version of Laplace correction is more commonly used in the literature than the second. Equation (4.2) shows the formula that was used in previous studies such as [Kohavi et al 1997, Provost and Domingos 2000].

$$P(w_j|c_k) = \frac{F(w_j, c_k) + f}{|c_k| + f|Categories|}, \quad (4.2)$$

where f is a penalty factor as previously mentioned, and $|c_k|$ denotes the total number of words in c_k .

Version II. The second version of Laplace correction [Agrawal et al 2000, Joachims 1996, Mitchell 1997] is different from the first in two ways: first, it does not have a variable f ; and second, it replaces $|Categories|$ in the denominator of Equation (4.2) with $|Vocabulary|$, the total number of distinct words in the training data.

$$P(w_j|c_k) = \frac{F(w_j, c_k) + 1}{|c_k| + |Vocabulary|} \quad (4.3)$$

Notice that in the place of a variable f , there is a constant 1 in Equation (4.3), which can not be optimized to achieve high accuracy values for different data sets. And since the value of $|Vocabulary|$ is usually very large (much greater than $|Categories|$),

it may dominate other values in Equation 4.3, and thus make the final result unreliable. In fact, when applied to the NBN data, *Version II* correction produces lower accuracy than no-match or *Version I* correction. This can be observed in Table 4.4 in the row where $f = 1$, although in our case, the differences are not very great because $|Vocabulary|$ is not extremely large (see Table 4.4). *Version II* of Laplace correction was further developed in [Agrawal et al 2000] as Lidstone’s law of succession, which will be discussed as Algorithm 3 later in this section. Therefore, when we talk about Laplace correction in this thesis, we mean *Version I*, and our attention for Laplace correction in this section is also focused on *Version I*.

According to [Kohavi et al 1997], the optimal value for f is $1/F(c_k)$. This is again application-dependent. Table 4.3 shows the relations between the optimal f values and the data sets. In our case, $1/F(c_k)$ is not optimal for f in any data set, on the contrary, the accuracy values in the first two columns under such f values are much worse than the optimal accuracies in bold.

minimum $F(c)$	1	10	50	100	200
average $ c $	1,318	1,635	3,045	4,411	9,088
$ Categories $	70	56	23	13	4
$f = 1$	8.01%	16.70%	42.86%	59.81%	90.77%
$f = 0.1$	12.84%	50.80%	70.27%	78.56%	91.24%
$f = 0.01$	39.17%	62.67%	72.91%	79.26%	90.93%
$f = 0.001$	57.78%	62.86%	72.91%	78.69%	90.62%
$f = 0.0001$	61.31%	62.00%	72.22%	77.92%	90.31%
$f = 0.00001$	60.93%	61.22%	71.20%	77.47%	90.15%
$f = 1/F(c_k)$	7.88%	45.17%	71.34%	79.04%	90.72%

Table 4.3: Laplace Correction – Accuracy Comparison of Top-level Hierarchical Classification with Different f Values, best accuracies in bold

By observing Table 4.3, we see that the optimal f value increases as the number of categories decreases. Again, this is what we would expect, because both f and $|Categories|$ terms are in the denominator in Equation (4.2), the smaller $|Categories|$

is (and thus the greater $\frac{1}{|Categories|}$), the smaller $\frac{1}{f}$ value we would need, and thus the greater f .

Algorithm 3: Lidstone’s law of succession

As mentioned in the previous section, *Version II* of Laplace correction is not effective owing to its lack of a variable. Therefore an intuitive approach would be to add a variable to Equation (4.3) to make the method adaptable to different data sets. Lidstone’s law of succession is such a method that was developed in [Agrawal et al 2000].

$$P(w_j|c_k) = \frac{F(w_j, c_k) + f}{|c_k| + f|Vocabulary|} \tag{4.4}$$

Note that when $f = 1$, Equation (4.4) is equivalent to Equation (4.3).

According to [Agrawal et al 2000], the optimal value for f is $1/F(c_k)$, the same as that for Laplace correction in Equation (4.2). Again, as shown in Table 4.4, this is not true for the NBN data set. It was also reported in [Agrawal et al 2000] that the overall accuracy of Lidstone’s law of succession is higher than that of Laplace correction. By comparing Tables 4.3 and 4.4, we see that this is true for most NBN subsets; the only exception is found in the subset with minimum $F(c) = 200$.

minimum $F(c)$	1	10	50	100	200
average $ c $	1,318	1,635	3,045	4,411	9,088
$ Vocabulary $	12,266	12,188	9,924	8,853	6,427
$f = 1$	54.88%	55.27%	72.97%	78.94%	90.67%
$f = 0.1$	63.24%	63.49%	73.07%	79.29%	91.08%
$f = 0.01$	62.95%	63.26%	72.89%	78.65%	90.72%
$f = 1/F(c_k)$	60.60%	60.78%	72.06%	79.01%	90.77%

Table 4.4: Lidstone’s Law of Succession – Accuracy Comparison of Top-level Hierarchical Classification with Different f Values, best accuracies in bold

It can be observed in Table 4.4 that, in comparison with the previous two smoothing methods, the optimal f value in Lidstone’s law of succession is very stable. This

is because the change of $|Vocabulary|$ over the five data sets is not as big as that of $F(c_k)$ and $|Categories|$, and thus f does not need to be adjusted much. For instance, the $\frac{1}{|Categories|}$ values for column 1 and 5 in Table 4.3 are $\frac{1}{70} \approx 0.014$ and $\frac{1}{4} = 0.25$ respectively, whereas the $\frac{1}{|Vocabulary|}$ values for column 1 and 5 in Table 4.4 are respectively $\frac{1}{12,266} \approx 1.6 \times 10^{-4}$ and $\frac{1}{6,427} \approx 0.9 \times 10^{-4}$. The difference between the former two values is much greater than that between the latter two.

Comparing Tables 4.3 and 4.4, we can see that, in general, we need a smaller f value in Laplace correction than in Lidstone’s law of succession. This is because the value of $|Categories|$ in Equation (4.2) is much smaller than that of $|Vocabulary|$ in Equation (4.4), and thus a smaller f value is needed for Laplace correction to penalize the non-observed words in a category.

4.1.2 Effect of Data with Different Sizes

The accuracies of the Naïve Bayes classifier reported in Section 4.1.1 are not very good. The only practically acceptable results were obtained on the data set where the minimum $F(c_k)$ is 200, for which the accuracy can reach 90%. But since there are only four such categories in the data set, the classifier will not be very useful in real life.

We suspect that such poor effectiveness of the Naïve Bayes classifier owes to the limitations of the training data. Recall that, up to now, all the experiments were done on the data of the NBON system of the last two years, which has a large number of sparse categories (see Table 3.3 on page 34). If we can get the NBON data for more years, there is a chance that we will obtain more categories with many tenders, of which the classifier is more confident in predicting, and thus makes the work more useful. A second expectation of getting more data is to cover more categories. Recall that in the original two-year data, 2 out of the 72 top-level categories were not covered, and as we have discussed in Section 3.5.3, non-coverage is a potential problem when the classifier is used in real life. Therefore, first another two years’ data from 1 January

1999 to 31 December 2000, and then another four years' data from 1 January 1995 to 31 December 1998, were obtained and added to the original data set for training and testing the Naïve Bayes classifier.

Table 4.5 shows some characteristics of the three data sets, the original last-2-year data, the last-4-year data, and the last-8-year data. The numeral pair in each position represents $|Tenders|$ and $|Categories|$ for each NBON subset, and the labels in the first column denote the different subsets. For example, under 2-year data, it says (in the row of “Total”) that there are all together 4,822 tenders classified in 70 top-level categories, and among which (in the row of “ $TF(c) \geq 10$ ”), there are 4,763 tenders labeled with 56 categories, each containing at least 10 tenders.

	2-year data 1 Jan 01 - 30 Nov 02	4-year data 1 Jan 99 - 30 Nov 02	8-year data 1 Jan 95 - 30 Nov 02
<i>Total</i>	4,822/70	9,720/71	25,020/72
$F(c) \geq 10$	4,763/56	9,686/62	25,015/69
$F(c) \geq 50$	3,743/23	9,167/44	24,583/54
$F(c) \geq 100$	3,123/13	7,831/26	24,278/50
$F(c) \geq 200$	1,945/4	6,167/13	21,955/34
$F(c) \geq 500$	1,457/2	3,408/3	14,695/11
$F(c) \geq 1,000$	0/0	2,890/2	8,713/3
$F(c) < 10$	59/14	34/9	5/3
$F(c) \leq 3$	12/7	11/6	5/3
$F(c) = 1$	4/4	2/2	2/2

Table 4.5: Characteristics of Three NBON Data Sets

It can be observed in Table 4.5 that our two expectations in getting more data were both satisfied. The number of sparse categories dropped, and in the 8-year data, all the 72 top-level categories were covered. Now we want to find out whether such data give us better accuracies, which was the reason why we wanted to get them in the first place. Table 4.6 shows the accuracy comparison of the different subsets in the three NBON data sets. To make things easier, Lidstone’s law of succession with $f = 0.1$ is the smoothing method used in these experiments.

	2-year data 1 Jan 01 - 30 Nov 02	4-year data 1 Jan 99 - 30 Nov 02	8-year data 1 Jan 95 - 30 Nov 02
$F(c) \geq 1$	63.24%	70.21%	70.41%
$F(c) \geq 10$	63.49%	70.41%	70.43%
$F(c) \geq 50$	73.07%	72.41%	71.16%
$F(c) \geq 100$	79.29%	77.62%	71.59%
$F(c) \geq 200$	91.08%	83.15%	74.45%
$F(c) \geq 500$		92.77%	84.93%
$F(c) \geq 1,000$			93.44%

Table 4.6: Accuracy Comparison of Top-level Hierarchical Classification on Three Data Sets

The results we obtained for the subsets consisting of categories with many tenders are disappointing. For instance, we achieved an accuracy of 91.08% for the 2-year data set consisting of four categories with more than 200 tenders each, and we would expect to improve such accuracy by getting more tenders in each of the four categories with more data. But the accuracy for such categories was actually dropped, because at the same time, we have got more categories in the new data sets: the numbers of such categories become 13 and 34 respectively in the 4-year and 8-year data sets. The increase of $|Categories|$ has resulted in the decrease of accuracy. This also verified our relationship analysis of accuracy and the number of categories in Section 3.5.1. Besides, even when $|Categories|$ was preserved and $|Tenders|$ increased, the accuracy did not increase much. For example, with the 2-year data, we obtained the best accuracy of 91.08% on the 4-category subset with $F(c) \geq 200$, and with the 4-year and 8-year data, the total numbers of categories in the best subsets have actually dropped from 4 to 3, and at the same time, the minimum $F(c)$ has greatly increased to 500 and 1,000 respectively, but disappointingly, the accuracies obtained were only 92.77% and 93.44%, which are very limited increases over 91.08%. This implies that the best accuracy that a Naïve Bayes classifier can achieve on the NBON data may not change much, no matter how much more historical data we get.

The good news is that, by observing the first two rows in Table 4.6, we see that the accuracies for the sparse data have greatly increased by approximately 7 percentage points. Looking back at Table 4.5, we can see that the number of categories with less than 10 tenders has dropped from 14 in the 2-year data to 9 and 3 respectively in the 4-year and 8-year data, and the number of very sparse categories has also dropped (see the last two rows). Such small improvement on the worst subsets is obviously more beneficial than the dramatic increase on the minimum $F(c)$ in the best subsets. This is actually better than what we expected, because it is more desirable that the classifier can work on all the categories rather than only on a small subset of the categories, no matter how well it can work on them.

4.2 Non-Bayesian Text Classification Techniques

We have shown in Section 4.1.2 that the effectiveness of a Naïve Bayes classifier on the NBON data is not likely to be improved much beyond the current level. The reason we have originally chosen Naïve Bayes as the classification technique for our application was that Naïve Bayes was reported by many researchers to be a technique both effective and practical. We have shown in Section 4.1.1 that some statements (e.g. the optimal f value) that have been proven to be true for other applications did not apply for the NBON data. This may also imply that Naïve Bayes may not be a good technique for our application just because it is for many other applications. For the purpose of a parallel comparison, we have implemented three equally practical (meaning simple to implement and straightforward to understand) approaches, two of which were reported in other studies to be equal to or better than Naïve Bayes in terms of accuracy.

In this section, the accuracies of the three non-Bayesian techniques are compared with that of the Naïve Bayes classifier using the 2-year NBON data. To make things easier, when comparing with other approaches, the accuracy results for Naïve Bayes

are represented by the top-level hierarchical classification using Lidstone’s law of succession with $f = 0.1$.

4.2.1 Strong Predictor Classifier: a Modified TF Classifier

Let us first define a baseline algorithm for the purpose of comparison. In traditional IR, TF is a very basic measure of relevance. Owing to its limitations, it is rarely used on its own. Our baseline algorithm is based on TF , and slightly modifies it to avoid its shortcomings.

The obvious problem with TF is that it does not take into account the case where a word occurs in many categories, and thus is not a good predictor for any category. Such a situation is taken into consideration by IDF in the TF-IDF classifier, which is our second non-Bayesian approach that will be discussed in Section 4.2.2. Our modification of the TF method avoids the necessity of an IDF term by adding an extra feature selection process during the training phase to screen out the words with high frequencies in many categories. We say a word w is a feature of a category c , if w was observed less than three times in the categories other than c . We call such features “strong predictors” (SP), implying that they are strong in predicting a category, because they are rarely observed in other categories. We then go further by restricting $F(w) = F(w, c)$, meaning that w was observed nowhere but in c . We call such features “unique predictors”, as a stricter form of SP, because they are unique to a certain category. Table 4.7 show a few top-level goods categories and some of their unique predictors.

Notice that there are some invalid features in the categories, such as French words and spelling errors. Such features do not contribute much in predicting a category, because they, fortunately, do not occur much in the data sets. For the same reason, they do not adversely affect the classification, either.

In the testing phase, the SP score of a category c is simply computed as the sum

GSIN 71	Furniture
	947 unique features, such as: <i>bookshelf, sheetself, fixedheight, mohagony, blanche, modèle, peuvent</i>
GSIN 72	Household and Commercial Furnishing and Appliances
	56 unique features, such as: <i>drape, blind, china, entryway, sqft, rideaux, verticaux</i>
GSIN 24	Tractors
	2 unique features: <i>crate, outfront</i>

Table 4.7: Example Top-level Goods Categories and Their Unique Predictors

of all the $TF(w, c)$ terms as shown in Equation (4.5).

$$score(c) = \sum_{w \in T} TF(w, c), \quad (4.5)$$

where T is a tender in the testing data, and the $TF(w, c)$ term is equivalent to our $F(w, c)$ term in the previous sections. Since each feature in a category occurs rarely elsewhere, the chance is great that $TF(w, c) = 0$. Zero count is a big problem for Naïve Bayes classifiers, for which $score(c)$ is the product of all the $P(w, c)$ terms, and a single zero count will cause the score to be zero. But this is not significant for a SP classifier, because its $score(c)$ is calculated as the sum over all the $TF(w, c)$ terms.

The decision rule for the SP classifier is thus given as follows:

$$c_{best} = \max_c \sum_{w \in T} TF(w, c) \quad (4.6)$$

Table 4.8 shows the comparison results of the accuracies of Naïve Bayes and the two SP classifiers.

It can be seen that the accuracies of the Naïve Bayes classifier is much better than that of the SP classifiers. One main reason that the SP classifiers are not very effective is that, by keeping only the strong predictors as the features for each category, a lot of useful information in the category has been lost. Especially with “unique predictors”, if a word occurred many times in a category and only once

minimum $F(c)$	1	10	50	100	200
total number of categories	70	56	23	13	4
categories with strong predictors	70	56	23	13	4
categories with unique predictors	68	56	23	13	4
SP - unique predictors	41.51%	42.06%	52.67%	63.72%	78.40%
SP - strong predictors	49.05%	49.10%	61.04%	68.81%	85.41%
Naïve Bayes	63.24%	63.49%	73.07%	79.29%	91.08%

Table 4.8: Accuracy Comparison of the SP and Naïve Bayes Classifiers for NBON Data

elsewhere, it was not recognized as a strong predictor, although it was actually very “strong” in predicting such category.

4.2.2 TF-IDF Classifier

TF-IDF is a traditional algorithm originally used in IR to reflect the relations between terms and documents. An IR system builds a term-document matrix that looks like Table 4.9 to record the frequencies of all the terms in all the documents in its database.

	d_1	d_2	d_3	...	d_k
t_1	$TF(t_1, d_1)$	$TF(t_1, d_2)$	$TF(t_1, d_3)$...	$TF(t_1, d_k)$
t_2	$TF(t_2, d_1)$	$TF(t_2, d_2)$	$TF(t_2, d_3)$...	$TF(t_2, d_k)$
t_3	$TF(t_3, d_1)$	$TF(t_3, d_2)$	$TF(t_3, d_3)$...	$TF(t_3, d_k)$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
t_j	$TF(t_j, d_1)$	$TF(t_j, d_2)$	$TF(t_j, d_3)$...	$TF(t_j, d_k)$

Table 4.9: Term-Document Matrix for TF-IDF

TF-IDF uses term weights to determine the relations between a term and a document. The weight of a term (i.e. a word) in a document is jointly determined by its term frequency in the document and its inverse document frequency in all the documents. When used in TC, the documents (i.e. tenders in our case) are replaced

by the categories, which are the sets of the documents that are labeled with such categories. TF-IDF is better than a simple TF approach in the sense that it introduces an *IDF* term to penalize the ubiquitous words that do not tell much about any category. The logic behind the TF algorithm discussed in Section 4.2.1 says that a word is a good representative of a category if it occurs many times in the category; that behind TF-IDF says that, in addition, a word is not a good representative of a category if it also occurs in many other categories.

In TC, the definition of $TF(w)$ for a word w is the same as in IR, that of $DF(w)$ is the number of categories that contain w , and $IDF(w)$ is computed as the fraction of the total number of categories in the sample data divided by $DF(w)$, smoothed with logarithm.

$$IDF(w) = \log\left(\frac{|Categories|}{DF(w)}\right) \quad (4.7)$$

Notice that if a word w_j occurs in all the categories, $IDF(w_j) = 0$. This is typically useful in getting rid of the noise introduced by the stop words.

The weight of a word w in a category c is given by the product of its *TF* and *IDF* values:

$$\begin{aligned} Weight(w, c) &= TF(w, c) \cdot IDF(w) \\ &= TF(w, c) \cdot \log\left(\frac{|Categories|}{DF(w)}\right) \end{aligned} \quad (4.8)$$

As in the TF algorithm, the score of a category c is calculated as the sum over all the $Weight(w, c)$ terms.

$$score(c) = \sum_{w \in T} Weight(w, c) \quad (4.9)$$

The decision rule of TF-IDF is again to choose the category with the highest score.

$$c_{best} = \max_c \sum_{w \in D} Weight(w, c) \quad (4.10)$$

Table 4.10 is an accuracy comparison of TF-IDF and Naïve Bayes classifiers, which shows that Naïve Bayes is a better technique than TF-IDF for our application.

minimum $F(c)$	1	10	50	100	200
TF-IDF	46.70%	48.47%	61.98%	68.78%	87.68%
Naïve Bayes	63.24%	63.49%	73.07%	79.29%	91.08%
Naïve Bayes (without feature selection)	61.66%	61.81%	71.85%	77.63%	90.46%

Table 4.10: Accuracy Comparison of the TF-IDF and Naïve Bayes Classifiers for NBO Data

Notice that Naïve Bayes does not directly address IDF , which serves as a penalty for ubiquitous words in a TF-IDF classifier. Instead, Naïve Bayes puts a penalty on non-observed words to achieve the same effect on the balance between TF and IDF . As previously mentioned, one motivation of IDF is to ignore the influence of the stop words, which tend to appear in all the documents. But in our case, since the stop words were removed from the feature space, IDF is not very useful in this sense. To show that Naïve Bayes is effective even when stop words are not removed beforehand, we did the experiments on the same data sets and show the accuracies in Table 4.10, which are not much worse than the ones with 456 stop words removed. We will show in Section 4.3.1 that the size of stop word lists do not have much effect on the Naïve Bayes classifier.

4.2.3 WIDF Classifier

Tokunaga and Iwayama [Tokunaga and Iwayama 1994] pointed out a problem with $IDF(w)$, which is that if a word is contained in all the categories, no matter what its frequency distribution is like, the weight of such a word in all the categories will be zero, because the $IDF(w)$ value is zero. The improperness of such weight computation method was illustrated by an example in [Tokunaga and Iwayama 1994], which is given in Table 4.11.

In the example, both $IDF(w_x)$ and $IDF(w_y)$ are equal to 0, because they both occur in all the categories, and thus both words will get a zero weight for all the

	c_1	c_2	c_3	c_4	c_5
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
w_x	2	50	3	2	4
w_y	3	2	3	2	3
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Table 4.11: A TF-IDF Example with a 5-Category Data Set

categories, although the frequency distributions of w_x and w_y are very different. The reason for this situation is that *IDF* considers only whether or not a word is contained in a category, but not its frequency in the category [Tokunaga and Iwayama 1994].

A Weighted Inverse Document Frequency (WIDF) classifier was developed in [Tokunaga and Iwayama 1994] to overcome the problem of TF-IDF by taking into account the frequency of the words.

$$WIDF(w, c) = \frac{TF(w, c)}{\sum_{i=1}^n TF(w, c_i)} \quad (4.11)$$

Now, the WIDF values of the two words in the above example would be different. For instance,

$$WIDF(w_x, c_2) = \frac{50}{2 + 50 + 3 + 2 + 4} = \frac{50}{61} \approx 0.82, \quad (4.12)$$

and

$$WIDF(w_y, c_2) = \frac{2}{3 + 2 + 3 + 2 + 3} = \frac{2}{13} \approx 0.15. \quad (4.13)$$

Since WIDF has already taken $TF(w, c)$ into account, the weight of a word w in a category c is simply its WIDF value.

$$\begin{aligned} Weight(w, c) &= WIDF(w, c) \\ &= \frac{TF(w, c)}{\sum_{i=1}^n TF(w, c_i)} \end{aligned} \quad (4.14)$$

The computation of $score(c)$ and the decision rule for WIDF are the same as that for TF-IDF in Equations (4.9) and (4.10) respectively.

Table 4.12 shows the accuracy comparison of the Naïve Bayes and WIDF classifiers for the NBON data.

It was reported in [Tokunaga and Iwayama 1994] that the accuracy of WIDF is better than that of TF-IDF by approximately 5%. Our experimental results have verified this statement, as can be seen in Table 4.10 and 4.12; the WIDF improves accuracies by about 2 to 12 percentage points.

It was also reported in [Tokunaga and Iwayama 1994] that the accuracies for TF-IDF and Naïve Bayes classifiers are comparable, whereas that for WIDF classifier are better. From Table 4.12 we can see that the accuracies for Naïve Bayes is better than that for WIDF.

minimum $F(c)$	1	10	50	100	200
WIDF	59.17%	59.43%	69.44%	75.06%	89.28%
Naïve Bayes	63.24%	63.49%	73.07%	79.29%	91.08%

Table 4.12: Accuracy Comparison of WIDF and Naïve Bayes Classifiers for NBON Data

4.3 Natural Language Processing Techniques

We have shown in Section 4.2 that Naïve Bayes is the optimal classification technique for the NBON system. In this section, we look at different ways to process the data with some NLP variations that may help to improve the effectiveness of a Naïve Bayes classifier.

In Chapter 3, we used two NLP techniques to process the NBON data, pruning the stop words and stemming the content words. In this section, some additions and variations on feature selection and stemming are discussed, and experiments are carried out to compare with the previous methods. As in Section 4.2, all the experiments in this section were carried out on the 2-year NBON data using Lidstone’s

law of succession with the penalty factor $f = 0.1$.

4.3.1 Feature Selection Methods

Pruning Words of Low Frequency

In Chapter 3, only one feature selection method was used, which is to prune words of high frequency (i.e. stop words). Another commonly used feature selection method is to prune words of low frequency. In this method, a word is only considered as a feature if it occurs at least a times in the training data (e.g. $a = 3$ in [Joachims 1996]). There are two main potential advantages of pruning words of low frequency. One is that most spelling errors will be removed since they, hopefully, do not occur identically many times in the training data. The other is to cut down the feature space and thus speed up the classification process. [Joachims 1996]

In this section, in addition to pruning words of high frequency using a stop list, we also prune words that occur less than 3 times in the training data. Table 4.13 shows the accuracy comparison of the Naïve Bayes classifier using different feature selection methods.

minimum $F(c)$	1	10	50	100	200
high frequency	63.24%	63.49%	73.07%	79.29%	91.08%
high and low frequency	60.91%	61.13%	70.64%	76.60%	90.10%

Table 4.13: Accuracy Comparison of Naïve Bayes Classifiers using Different Feature Selection Methods

Disappointingly, the results obtained for the new feature selection method are much worse than those of the previous trial. Before looking into the reason behind this failure, let us first try something else. Recall that in Section 4.2.1, we discussed the potential reasons for the poor effectiveness of the SP classifier, and one of which is the “abnormal” words, such as the spelling errors, being considered as the strong

predictors for a category. Since one of the advantages of pruning infrequent words is the chance of getting rid of most spelling errors, we want to see whether this is helpful in improving the effectiveness of the SP classifier.

Table 4.14 shows the accuracy comparison of SP classifiers using different feature selection methods. SP refers to the “strong predictors” with the frequency of the predictors less than 3 times elsewhere, and UP stands for “unique predictors”, the ones that were observed nowhere else.

minimum $F(c)$		1	10	50	100	200
SP	high frequency	49.05%	49.10%	61.04%	68.81%	85.41%
	high and low frequency	40.19%	40.29%	52.25%	60.93%	80.05%
UP	high frequency	41.51%	42.06%	52.67%	63.72%	78.40%
	high and low frequency	29.09%	30.08%	39.97%	51.38%	69.23%

Table 4.14: Accuracy Comparison of SP Classifier using Different Feature Selection Methods

Again, the results are disappointing. Comparing Tables 4.13 and 4.14, we can see that the adverse influence of pruning words of low frequency is much greater in the SP classifiers than in Naïve Bayes. The reason for this difference can be easily imagined, that is by excluding infrequent words in the feature space, some real strong predictors are removed. This is also the reason for the worse accuracies obtained in Naïve Bayes classifier, where some useful features were omitted. Besides, owing to the nature of the NBON system, the documents supplied to the system were all official tenders, and thus it should be safe to assume that there are few spelling errors in the data. Therefore, a main advantage of pruning infrequent words does not hold for our application, since there were not many spelling errors to be removed.

Pruning Words of High Frequency Using Different Stop Lists

Since it seems a bad idea to look for new feature selection method for this particular application, let us now go back to the previous feature selection method to see whether

any improvement can be carried out.

Recall that the stop list used in Chapter 3 contains 456 words. This is a relatively big stop list. An obvious argument against big stop lists is that they may contain some useful information that has been ignored. For instance, in our 456-word stop list, there are words like “indoors”, ”slung”, etc., which do have “content”, and thus could be predictors for some categories.

In this section, we use a much smaller stop list that contains only 36 words as the basis for our feature selection process. The 36 stop words are listed in Table 4.15. It can be seen that they are the true stop words in terms of “no content”.

a	about	an	and	are	at
be	but	by	her	his	i
in	is	it	me	mine	my
nor	not	on	or	our	over
the	their	they	under	was	we
were	with	within	without	you	your

Table 4.15: A 36-Word Stop List [Stop List]

Table 4.16 shows the accuracy comparison of the two feature selection bases. We

minimum $F(c)$	1	10	50	100	200
456-word stop list	63.24%	63.49%	73.07%	79.29%	91.08%
36-word stop list	62.37%	62.44%	72.43%	78.43%	90.72%
0-word stop list	61.66%	61.81%	71.85%	77.63%	90.46%

Table 4.16: Accuracy Comparison of Naïve Bayes Classifications using Different Stop Lists

see that the accuracies for the small stop list are slightly worse than that of the big one. An argument against small stop lists is that they tend to be too small to remove the noises (such as “also”, “for”, etc.) in the training data, which was the original motivation of having a stop list. Recall that, in Section 4.2.2, we experimented the

Naïve Bayes classifier without removing stop words in the tender, and the accuracies are even worse as shown in Table 4.16, because the size of the stop word list, which is 0 in this case, is smaller.

Table 4.17 shows one example of the errors made by a smaller stop list. The first row shows the original tender, below it the correct (on the left) classification and processed text by a Naïve Bayes classifier using a 456-word stop list, and the wrong ones (on the right) by a Naïve Bayes classifier using a 36-word stop list. The words in *italic* are the stop words that were not recognized and removed from the text by the classifier.

One tablestand for the fax machine PROP50452MG - Gray	
Correct classification	Wrong classification
456-word stop list	36-word stop list
71 - Furniture	70 - General Purpose Automatic Data Processing Equipment (including Firmware), Software, Supplies and Support Equipment
tablestand fax machine prop50452mg gray	<i>one</i> tablestand <i>for</i> fax machine prop50452mg gray

Table 4.17: An Example Error Made by a 36-word Stop List

4.3.2 Stemming Algorithms

Now that the argument for feature selection is settled, let us look at the other NLP technique used in this thesis, stemming.

Recall that Porter Stemmer that was introduced in Section 3.3 attempts to remove about 60 English suffixes, and Lovins stemmer, the other frequently used stemming algorithm in NLP, removes more than 290 suffixes. An intuitive question would be whether the more suffixes removed, the better, or the other way around; or maybe a stemmer should not be used at all. In order to answer such questions, let us look into

the two popular stemming algorithms and try to identify some common problems and advantages and disadvantages of each.

Problem 1: stemming is ignorant of semantics.

Stemming is a pure pattern recognition process, and the meanings of the words being stemmed are not considered [Riloff 1995]. A large number of morphologically similar words come from derivations. Some words were derived from the same “root” word, but their meanings can be very different. Recall what polysemy means, i.e. a word may have multiple distinct meanings, and thus it is not surprising that the words derived from the different meanings of the same word should be semantically different from each other.

Consider the following groups of words:

art: arts, artless

custom: customs, customer, customary, customize

capital: capitalism, capitalize

The words on the left are the root words and the ones on the right are some variations and derivations of the same word. It can be seen that the meanings of some variations and derivations of the same word can be totally different, or only slightly related. But such differences are ignored by stemming. For example, both Porter and Lovins stemmers will consider “capitalize” to be the verb form of “capital” and remove the “ize” suffix, but the two words may not mean the same thing, although sometimes they do. The “capital” here may refer to the government residence of a country or province, and “capitalize” may mean to write in capital letters.

Porter stemmer is better than Lovins stemmer in the sense that it reduces the number of removable suffixes for derivations, and thus decreases the mistakes made on the assumptions about the semantic similarities of the derived words.

Notice that some nouns may have different meanings for their singular and plural forms. In our example, “customs” may refer to import duties, and thus have nothing

to do with the “custom” of, for example, a social group. Same happens to “art” and “arts”. Such situations can not be taken care of by any rule-based stemming algorithm.

Problem 2: stemming may fail to recognize some derivations.

Opposite to Problem 1, some words that are both morphologically and semantically similar to each other may not be reduced to the same root form by some stemming algorithms. Since Porter stemmer has a smaller number of removable suffixes than Lovins stemmer, it is therefore easy to imagine that Porter stemmer will fail to recognize some semantically closely related derivations of the same word. In addition, Porter stemmer does not recognize root changes, which happen in many derivations. For example, “deride” and “derision” are simply the verb and noun forms of the same concept, but Porter stemmer will not reduce them to the same root. In this case, Lovins stemmer is better, because it does have a transformation rule that will recognize the root change from “deride” to “derision”, and thus with Lovins stemmer, the two words will end up with the same root.

Problem 3: stemming is rule-based.

Stemming algorithms are typically rule-based. Both Porter and Lovins stemmers base their computations on a set of linguistic rules, and therefore share all the problems suffered by rule-based approaches.

First, rules typically coexist with conditions under which the rules apply. Recognition and analysis of such conditions can be complicated for machines, if not for humans, to carry out. Second, rules are not dynamic, whereas natural languages are, and it is impractical to frequently update the rules. Although linguistic rules of a natural language are, fortunately, not changing all the time, new words may not always follow such rules. This leads to the third problem with rule-based approaches, that is, every rule has exceptions. Stemming algorithms are typically helpless in the

case of irregular words. A possible solution to irregularity is to add some special rules to the existing rule space. Such an approach, if not impossible to carry out (because such rules are not easy to find owing to the fact that all the irregular words are not irregular in the same way, otherwise they would make up a new rule), would be very expensive. Besides, some special rules may violate the regular ones.

The original motivation of stemming was to address synonymy. But we have shown that it turns out to sacrifice polysemy for this improvement. In the literature, there are opposite opinions about the effectiveness of stemming, some (e.g.[Harman 1991]) showed that stemming has no effect on improving recall and precision in IR, whereas some (e.g.[Krovetz 1993]) reported significant improvement. In the following two subsections, we experiment two alternative design choices, restrictive stemming and no stemming.

Restrictive Stemming

The reason why we did not choose Lovins stemmer was that it removes too many suffixes. If we view Lovins stemmer as extreme stemming, the Porter stemmer is comparatively restrictive, because it does not attempt to remove many suffixes, but removes a restrictive subset of the English suffixes. We have shown that both stemmers have advantages and drawbacks. In recent years, more researchers tend to believe that restrictive stemming is preferable for many applications. Although restrictive in comparison with Lovins stemmer, the Porter stemmer still removes as many as about 60 suffixes, some of which may be important in preserving the original meaning of a word. For instance, the Porter stemmer removes suffixes for conjugated verbs, which may imply some information that should not be ignored. For example, we all know that a “developing country” is very different from a “developed” country. The progressive and past tense conjugations of a verb typically represent the active and passive voices respectively, and thus shall be kept to preserve their original meanings.

Now it seems that the plural noun endings are the only comparatively safe suffixes to remove. Therefore our restrictive stemming algorithm only attempts to remove the “s” and “es” endings for English nouns. The algorithm implemented here is a modification of the standard Porter stemmer. It omits phases 2 to 5 which remove other kinds of suffixes, and only keeps the part of phase 1 that removes the plural noun endings. In addition, the Porter stemmer only includes two cases for the “es” endings, “sses” and “ies”; in our restrictive stemmer, three more such cases were added, “xes” and “ches”. Notice that irregular endings such as “i” (as in “alumni”) can not be recognized by either stemmer. Table 4.18 shows the accuracy comparison of Naïve Bayes classifier with Porter and restrictive stemming algorithms. We can see that restrictive stemming is better than Porter stemming for the NBON data set.

minimum $F(c)$	1	10	50	100	200
Porter stemming	63.24%	63.49%	73.07%	79.29%	91.08%
Restrictive stemming	63.49%	63.72%	73.72%	79.55%	91.65%

Table 4.18: Accuracy Comparison of Naïve Bayes Classifiers using Different Stemming Algorithms

Table 4.19 shows an example of misclassification by over stemming. Similar to Table 4.17, the first row shows the original tender, and below it the correct and wrong classifications and processed texts. The words in *italic* are the ones over stemmed by the Porter stemmer.

Note that some mistakes made in Porter stemmer cannot be avoided by our stemming algorithm either. For instance, there is no easy way to tell a stemmer that “calculus” is not the plural form of “calculu”, unless a machine readable dictionary is involved in the stemming process. Such approach was taken by Krovetz in the development of a derivational stemmer [Krovetz 1993].

Weighing Dishes No.02-202A Hexagonal polystyrene (6 pkg/case)	
Correct classification	Wrong classification
Restrictive Stemming	Porter Stemming
66 - Instruments and Laboratory Equipment	84 - Clothing, Individual Equipment and Insignia
weighing dishe no.02-202a hexagonal polystyrene pkg/case	<i>weigh</i> dishe no.02-202a <i>hexagon</i> polystyrene pkg/case

Table 4.19: A Misclassification Example of the Porter Stemmer

No Stemming

We mentioned on page 63 that some researchers (e.g. [Harman 1991]) believe that stemming has no effect on improving accuracy at all. If this is true with our application, why should we go through the trouble to stem every word in the sample data? It was shown in [Riloff 1995] that “little words can make a big difference for text classification”. Such “little words” include plural nouns, the only kind of suffix we attempt to remove in our restrictive stemming algorithm. The argument in [Riloff 1995] about the difference between singular and plural nouns is that the former usually refer to a specific incident, whereas the latter often refer to the general types of incidents, and such differences may influence the relevancy of the words in some domains. Besides, as we have shown in problem 1 of stemming, some nouns change their meanings when transforming from singular to plural form. This suggests that even our humble stemming algorithm may have done too much in processing the text.

In this section, we compare the accuracies obtained by text operations with and without stemming. Such comparison results are shown in Table 4.20.

Despite of the suspicion many researchers have on the effectiveness of stemming, most IR systems still use it. We can see in our experiments that it is better not to use stemming at all than to use extreme stemming, and that restrictive stemming is the best choice to achieve better accuracy, although the influence of any kind of stemming is insignificant.

minimum $F(c)$	1	10	50	100	200
Porter stemming	63.24%	63.49%	73.07%	79.29%	91.08%
No stemming	63.17%	63.53%	73.21%	79.46%	91.49%

Table 4.20: Accuracy Comparison of Naïve Bayes Classifiers With and Without Stemming

4.4 Comparative Results and Error Analysis

In this section, we present a comprehensive comparison of the techniques implemented and the experiments in this thesis, and try to identify and analyze the errors made by the algorithms, and propose some potential solutions.

4.4.1 Comparative Results

Comparative Results of Different TC Techniques

Naïve Bayes is the main technique implemented in this thesis. In this chapter, three different algorithms of Naïve Bayes classifier that deal with zero counts were experimented and the accuracies were compared. In order to see whether Naïve Bayes is the optimal technique for the NBON system, we have implemented three non-Bayesian classifiers, namely, SP, TF-IDF and WIDF, and the experiment results have shown that a Naïve Bayes classifier achieves better accuracies.

Table 4.21 combines the accuracy comparisons previously shown in Sections 4.1 and 4.2, and thus includes all the TC techniques implemented in this thesis.

Comparative Results of Naïve Bayes Classification on Different Data Sets

Since we were not very satisfied with the accuracies obtained on the NBON data of the recent two years, we tried to run the Naïve Bayes classifier on two more data sets, the 4-year and 8-year NBON data, in order to see the effects of the sizes of hypothesis space and sample space on the effectiveness of Naïve Bayes classifier.

minimum $F(c)$	1	10	50	100	200
<i>Tenders</i>	4,822	4,763	3,743	3,123	1,945
<i>Categories</i>	70	56	23	13	4
<i>Vocabulary</i>	12,266	12,188	9,924	8,853	6,427
average <i>c</i>	1,318	1,635	3,045	4,411	9,088
SP - unique predictors	41.51%	42.06%	52.67%	63.72%	78.40%
SP - strong predictors	49.05%	49.10%	61.04%	68.81%	85.41%
TF-IDF	46.70%	48.47%	61.98%	68.78%	87.68%
WIDF	59.17%	59.43%	69.44%	75.06%	89.28%
Naïve Bayes - no-match	61.54%	62.00%	72.78%	79.65%	91.34%
Naïve Bayes - Laplace	61.31%	62.86%	72.91%	79.26%	91.24%
Naïve Bayes - Lidstone	63.24%	63.49%	73.07%	79.29%	91.08%

Table 4.21: Accuracy Comparison of all the Text Classification Algorithms in this Thesis

Table 4.22 selectively combines parts of Tables 4.5 and 4.6 and shows the best and worst accuracies obtained under different conditions in the three NBON data sets. Recall that all the accuracies were obtained using Lidstone’s law of succession with a penalty factor $f = 0.1$.

	2-year	4-year	8-year
	1 Jan 01 - 30 Nov 02	1 Jan 99 - 30 Nov 02	1 Jan 95 - 30 Nov 02
Best Accuracy	91.08%	92.77%	93.44%
<i>Minimum $F(c)$</i>	200	500	1,000
<i>Tenders</i>	1,945	3,408	8,713
<i>Categories</i>	4	3	3
Worst Accuracy	63.24	70.21%	70.41%
<i>Minimum $F(c)$</i>	1	1	1
<i>Tenders</i>	4,822	9,720	25,020
<i>Categories</i>	70	71	72

Table 4.22: Accuracy Comparison of Naïve Bayes Classifications on Three Different Data Sets I

It can be observed that the differences between the worst accuracies are greater than those of the best. In both cases, |*Categories*| did not change much in the

three data sets, but $|Tenders|$ greatly increased. At the same time, $Minimum F(c)$ remained the same for the worst accuracies, and greatly increased for the best. However, the accuracy increases are greater in the worst cases. This is preferable to increasing best case accuracy because worst case solutions are greater in number and are harder to resolve. At the same time, the limited increase we obtained in accuracy as the minimum $F(c)$ increases shows that no matter how many tenders are given, the Naïve Bayes classifier may not perform better on the NBON system than its current level.

Table 4.23 shows the accuracy increase obtained by working with the ranking method discussed in Section 3.6.3 on page 38.

	2-year 1 Jan 01 - 30 Nov 02	4-year 1 Jan 99 - 30 Nov 02	8-year 1 Jan 95 - 30 Nov 02
$ Tenders $	4,822	9,720	25,020
$ Categories $	70	71	72
$ Vocabulary $	12,266	19,086	30,291
average $ c $	1,318	2,736	5,913
Top 1 Accuracy	63.24%	70.21%	70.41%
Top 3 Accuracy	73.90%	79.58%	81.22%
Top 5 Accuracy	78.09%	83.18%	85.14%
Top 10 Accuracy	83.92%	87.54%	89.37%

Table 4.23: Accuracy Comparison of Naïve Bayes Classifications on Three Different Data Sets II

It is good to see that the accuracy can be close to 90% in the 8-year data when the top 10 ranked categories are returned to the users, who will have a chance to choose one (or more) for themselves.

Comparative Results of Different NLP Techniques

Besides the TC techniques, we have also tried some NLP techniques to process the NBON data before using them on the Naïve Bayes classifier. In Section 4.3, we

have further discussed two NLP techniques used in this thesis, feature selection and stemming, and some variations on each have been implemented and experimented with.

Table 4.24 shows the comprehensive accuracy comparison of all the NLP variations implemented in this thesis for the Naïve Bayes classifier. Again, all the accuracy values shown here were obtained using Lidstone’s law of succession with $f = 0.1$. We can see

minimum $F(c)$	1	10	50	100	200
high and low frequency, Porter stemming	60.91%	61.13%	70.64%	76.60%	90.10%
0-word stop list, Porter stemming	61.66%	61.81%	71.85%	77.63%	90.46%
36-word stop list, Porter stemming	62.37%	62.44%	72.43%	78.43%	90.72%
456-word stop list, Porter stemming	63.24%	63.49%	73.07%	79.29%	91.08%
456-word stop list, No stemming	63.17%	63.53%	73.21%	79.46%	91.49%
456-word stop list, Restrictive stemming	63.49%	63.72%	73.72%	79.55%	91.65%

Table 4.24: Accuracy Comparison of Naïve Bayes Classifications using Different NLP Techniques

that the accuracy differences between the NLP techniques are not significant. This also supported the hypothesis that the effect of NLP in TC is limited [Harman 1991]. Although a big stop list (with 456 words) with restrictive stemming does seem to be a good combination for our application.

4.4.2 Error Analysis

It seems that no matter what we try, the effectiveness of the Naïve Bayes classifier can not be increased much and it may not be enough to meet the practical requirements of the NBON system in terms of accuracy. Recall that the best accuracy we obtained for all the experiments is approximately 90%, which means that the error rate is as high as 10%, which is not practical for use in real world application. Therefore it is now time for us to look into the error data and try to find the reasons for the unsatisfactory effectiveness of the classifier.

Errors in Human Indexing

By observing the NBON data, we found that some tenders might not have been correctly labeled by human indexers. Such wrongly labeled tenders have become noise in the training data, and when they served as the source for testing, a correct prediction by the system has been considered wrong, and thus decreased the accuracy. We have mentioned at the beginning of this thesis that the motivation of this thesis is to replace human indexing with automatic or semi-automatic classification, because manual labeling is error-prone (and expensive). This statement has now been verified, that human indexers do make mistakes.

What we can do is to manually reexamine the data used for training the classifier and correct the wrong labels. Such examination can be done using some automatic processes. For instance, those tenders, whose labeling the system agreed with the original human indexing, need not be touched. Therefore effort can be saved and focused on the potentially wrongly labeled tenders. When the load of the relabeling work is not very intimidating, the chance of human mistakes can be reduced.

Foreign Words in the NBON Data

Owing to the fact that Canada has two official languages, English and French, and that New Brunswick is a bilingual province, there are a few tenders that were written in both English and French. Both NLP methods we used in this thesis, feature selection and stemming, are rule-based, and thus are language-dependent. A stop list and a stemmer for English will definitely not work for French. Besides, some French words might have been treated as strong predictors by the SP classifier. Since we did not include any language recognition procedure in our text operations, the classifier, which can not tell one language from another, has treated all the words as equal features. Noise from French data, although relatively rare, have definitely adversely affected the classification.

We need to introduce some strategy to recognize the languages being used. A simple approach would be to redesign the user interface and ask the users to write different languages in different places. Since English is more commonly used as the business language in the province of New Brunswick, a tender for the NBON system was never written in French only, we can use the English tenders for classification purposes, and French, if any, can exist for the display purpose only.

Inevitable Mistakes Made by NLP Tools

In this thesis, we used a stop list as the basis for our feature selection method. We have mentioned some arguments against both big stop lists and small ones in Section 4.3.1, a big stop list may contain some non-stop words for a particular application, and a small stop list may be too general to remove most noise in a tender. An alternative approach for feature selection is to use frequency counts to prune frequent or/and infrequent words, although sometimes it can be tricky to define “frequent” for a particular application.

We have discussed in Section 4.3.2 that what a stemmer can do to reduce a word to its stem is limited. One such limitation is reflected in irregularity, such as irregular verbs and nouns that take rare conjugation and plural endings. The other is the false recognition of a root part as an ending, such as “bus” being considered as the plural form of “bu”, and thus will not share the same stem as its plural form “buses” when it is stemmed to “bus”. A third is the tradeoffs between synonymy and polysemy reflected in stemming. Since there are advantages and disadvantages in any kind of stemming, it is a design choice, and we do not have one perfect solution to all the problems.

Chapter 5

Conclusion

In this thesis, we have taken a machine learning approach to automatically assign GSIN codes to the tenders of the NBON system. We have implemented different TC techniques and compared the accuracies achieved, and we have experimented with different NLP techniques to try to improve the accuracies.

5.1 Machine Learning Techniques for Text Classification

Among various machine learning techniques developed for TC tasks, we chose the Naïve Bayes classification method because of its simplicity and effectiveness. We have implemented a Naïve Bayes classifier to learn the characteristics of the categories from up to 8 years' historic data of the NBON system, and predict the category for an unseen tender using the learned knowledge.

We have tried three smoothing algorithms to deal with zero counts, namely, no-match, Laplace correction, and Lidstone's law of succession. According to the accuracy comparison of the three algorithms, we conclude that the difference between

them is trivial, which is around one percentage point (see Table 4.21). In most experimentation cases, Lidstone's law of succession achieved slightly better accuracies than the other two algorithms; in other cases, the no-match method performed better; and the accuracies of Laplace correction were always inbetween.

For parallel comparison, we have implemented three non-Bayesian classifiers that are as simple as Naïve Bayes, namely, SP (strong predictors, a modification of the TF algorithm), TF-IDF (term frequency - inverse document frequency) and WIDF (weighted inverse document frequency), and have compared their effectiveness with that of the Naïve Bayes classifier. The comparison results have shown that Naïve Bayes achieved much better accuracies than the other techniques. The differences between the accuracies of Naïve Bayes and that of the other techniques vary from 2 to 20 percentage points (see Table 4.21).

5.2 Natural Language Processing Techniques for the Naïve Bayes Classifier

For the purpose of the TC task, we have adopted two NLP techniques for the text operations on the training data: feature selection and stemming. In order to improve the effectiveness of the Naïve Bayes classifier, variations of the two NLP techniques have been carried out.

For feature selection, we have tried two approaches: one is pruning infrequent words, which decreased the accuracies of the Naïve Bayes classifier; the other is pruning words of high frequency using stop lists of different sizes, and it turned out that a bigger stop list (with 456 words) is better than a smaller one (with 36 words) in terms of accuracy improvement in the NBON system.

As for stemming, we have tried stemming using the standard Porter stemmer,

restrictive stemming using a modified Porter stemmer, and no stemming. Experimentation on the NBON data has shown that restrictive stemming achieved the best accuracies, no stemming worse, and Porter stemming the worst. This implies that heavy stemming can decrease the accuracies of a classifier, and therefore it would be better not to use a stemmer at all. When stemming is involved in text operations, a restrictive stemmer is a better choice.

The comparison results of different NLP techniques have shown that the effect of NLP on the accuracies of TC tasks is very small, only around 2 percentage points (see Table 4.24).

Chapter 6

Future Work

6.1 Improvement on the Reliability of the Data

In Chapter 4, we identified some errors in the human indexing of the NBON data that contributed to the unsatisfactory effectiveness achieved by the Naïve Bayes classifier. If we want to further improve the accuracies, the erroneous data must be either removed or corrected so that the classifier will not learn any wrong knowledge about the categories.

As we have discussed in Chapter 4, the improvement on the training data can be fulfilled using a combination of automatic and manual corrections. Alternatively, the erroneous data can be simply discarded to save effort spent on data correction. This will result in a decrease in the number of the training examples, and in addition, the identification of erroneous data can be time-consuming itself.

6.2 Recognition of French Tenders

We have shown in Chapter 4 that the existence of some French words in the NBON data has resulted in invalid features in some categories. In order to get rid of such “noise”, we must make sure that the Naïve Bayes classifier is trained on only one

language, because the NLP tools involved in the classification process are all language-dependent. When the classifier is used in the real world, we must guarantee that new English tenders are classified by their English version only, and maybe, eventually French by French. After being classified, they will serve as the training data to update the knowledge of the classifier, and such training data shall be again the English version of the tenders for classifying future English tenders.

Elimination of the French part of a tender is not an easy task when the tender is already included in the training data, because it will involve the identification of the beginning and the end of the French part. Therefore, identification of French versus English tenders is preferably done in the user interface, before they are forwarded to the classifier. It should be easy to redesign the user interface to require the users to write English and French in different text boxes, and pass the English tenders into the classifier, while keep the French tenders for displaying purpose only, until sufficient data is collected to train a French classifier, if necessary. A straightforward process of French recognition can be easily added to check the English tenders before forwarding them to the classifier.

The French recognition process can be as easy as to identify some typical French stop words such as “de”, “et”, etc. If such words are identified in the English text input box, the whole tender will be rejected and the user will be reminded to write French in a separate text box. Some threshold of the frequency of French words can be defined as the criteria for tender rejection, because a single occurrence of a French-like word (e.g. a French name) does not always imply that the whole tender is not written in English only.

6.3 Lower-level Hierarchical Classification

The accuracies achieved in Chapters 3 and 4 were based on the top-level hierarchical classification. Although we have assumed in Section 3.6.1 that it is sufficient to

classify tenders high in the hierarchy, it is preferable to classify both tenders and vendors as precisely as possible. In addition, by classifying tenders and vendors low in the hierarchy, we will have more flexibility in modifying the notification strategy when necessary.

We have achieved top-level hierarchical classification by simply using pattern recognition of the GSIN codes. For the purpose of lower-level hierarchical classification, we will need to build a tree-like structure for the hypothesis space of the Naïve Bayes classifier, so that the categories are internally connected by the parent-child relations, which will allow the classifier to go down the hierarchy to classify tenders under lower-level nodes.

Since we plan to involve users in the classification process (see Section 3.6.3), the accuracy of the classification on each level can be guaranteed, and thus we can safely go further down the hierarchical structure and simply repeat the classification process on each level until either the classifier can not go further down the hierarchy, or the user chooses to stop at a certain level of precision.

6.4 Classification of Vendors

In order to activate the automatic classification of vendors, we have suggested in Section 2.2.3 some modification on the user interface of the current NBON system. We will primarily need a place on the user interface that allows vendors to create their expertise documents or provide the tenders that they would like to see, based on which the classifier can assign GSIN codes to the vendors.

6.5 Classification of Services Tenders

In this thesis, the historic data used for training purpose is the “goods” tenders of the NBON system over the past years. When used in the actually application, the

classifier will definitely need to be able to deal with the tenders for all the three general groups, namely, “goods”, “services”, and “construction services”.

One advantage of machine learning techniques is domain independence, and therefore the implementation of the Naïve Bayes classifier in this thesis is independent of the group with which it is working. Such independence makes the expansion of our classifier to include all the three general groups an easy task. We can either build separate classifiers for the three groups, or use one classifier for all the three groups which identifies different groups by the patterns of their GSIN codes.

One problem is that, unlike the vendors, the purchasing agents do not tell the current system to which group their tenders belong. The result will be that the system does not know which classifier to use, or that the classifier ends up searching the categories in all the three groups, which will both increase the search time and decrease the accuracy. We have talked about redesigning the user interface for the vendors in Section 2.2.3, now we will need to redesign that of the tenders as well, so that the system knows with which super-level group it is working. This means, when a purchasing agent requests both goods and services, he or she will have to submit two separate tenders. In addition, when the request contains more than one item, he or she will have to separate them manually. This also applies to vendors when they sell products or services in more than one area.

Bibliography

- [Agrawal et al 2000] Agrawal, R., R. Bayardo and R. Srikant. 2000. Athena: Mining-based Interactive Management of Text Databases, *Extending Database Technology*.
- [Attias 2000] Attias, H. 2000. A Variational Bayesian Framework for Graphical Models, in *Advances in Neural Information Processing Systems 12*. MIT Press, Cambridge, MA
- [Blei et al 2003] Blei, D.M., M.I. Jordan and A.Y. Ng. 2003. Hierarchical Bayesian Models for Applications in Information Retrieval, *Bayesian Statistics 7*, pp. 25-43
- [Domingos 1996] Domingos, P. and M. Pazzani. 1996. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Proceedings of the 13th International Conference on Machine Learning* (pp. 105-112).
- [DVL/Verity] Defense Virtual Library / Verity Stop Word List.
http://dvl.dtic.mil/stop_list.html
- [Fukumoto and Suzuki 2002] Fukumoto, F. and Y. Suzuki. 2002. Manipulating Large Corpora for Text Classification, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, pp. 196-203.
- [Gruber 1993] Gruber, T.R. 1993. A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, 5(2):199-220.

- [Harman 1987] Harman, D. 1987. A Failure Analysis on the Limitations of Suffixing in an Online Environment, *Proceedings of the 10th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New Orleans, Louisiana, US.
- [Harman 1991] Harman, D. 1991. How Effective is Suffixing? *Journal of the American Society for Information Science*, 42(1):7-15
- [Jackson and Moulinier 2002] Jackson, P and I. Moulinier. 2002. *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*. John Benjamins Publishing Company, Amsterdam/Philadelphia
- [Joachims 1996] Joachims, T. 1996. *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*. Computer Science Technical Report CMU-CS-96-118, Carnegie Mellon University
- [Kohavi et al 1996] Kohavi, R., D. Sommerfield and J. Dougherty. 1996. Data Mining using MLC++: A Machine Learning Library in C++, in *Tools with Artificial Intelligence*, pp. 234-245. IEEE Computer Society Press
- [Kohavi et al 1997] Kohavi, R., B. Becker and D. Sommerfield. 1997. *Improving Simple Bayes*. Data Mining and Visualization Group, Silicon Graphics, Inc., Mountain View, CA
- [Kononenko 1991] Kononenko, I. 1991. Semi-naive Bayesian Classifiers. In *Proceedings of the 6th European Working Session on Learning*, 206-219.
- [Krovetz 1993] Krovetz, R. 1993. *Viewing Morphology as an Inference Process*. Technical Report UM-CS-1993-036, University of Massachusetts, Amherst, MA.
- [Lovins 1968] Lovins, J. 1968. Development of a Stemming Algorithm. *Machine Translation and Computational Linguistics*, Vol. 11, pp. 22-31

- [Manning and Schütze 1999] Manning, C.D. and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press
- [Mitchell 1997] Mitchell, T.M. 1997. *Machine Learning*. The McGraw-Hill Companies, Inc.
- [NBON] New Brunswick Opportunities Network. <https://nbon-rpanb.gnb.ca>
- [Pedersen 1999] Pedersen, T. 1999. *Search Techniques for Learning Probabilistic Models of Word Sense Disambiguation*, in *AAAI Spring Symposium on Search Techniques for Problem Solving Under Uncertainty and Incomplete Information*. Palo, Alto, CA
- [Porter 1980] Porter, M.F. 1980. The Porter Stemming Algorithm.
<http://www.tartarus.org/martin/PorterStemmer>
- [Provost and Domingos 2000] Provost, F and P. Domingos. 2000. *Well-trained PETs: Improving Probability Estimation Trees*. CDER Working Paper No. 00-04-IS, Stern School of Business, NYU.
- [Riloff 1995] Riloff, E. 1995. Little Words Can Make a Big Difference for Text Classification, in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 130-136
- [Sebastiani 2002] Sebastiani, F. 2002. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, Vol.34, No.1, March 2002, pp.1-47
- [Stop List] http://www.sliceware.com/documentation/Trans_Table/Stop_List.htm
- [Tesauro 1995] Tesauro, G. 1995. Temporal Difference Learning and TD-gammon. *Communications of the ACM*, 38(3), 58-68.

- [Tokunaga and Iwayama 1994] Tokunaga, T. and M. Iwayama. 1994. *Text Categorization Based on Weighted Inverse Document Frequency*. Technical Report 0918-2802, Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan
- [Woods et al. 2000] Woods, W.A., L.A. Bookman, A. Houston, R.J. Kuhns, P. Martin and S. Green. 2000. *Linguistic Knowledge can Improve Information Retrieval*. Technical Report No. TR-99-83, Sun Microsystems Laboratories, Burlington, MA.
- [Yang and Pedersen] Yang, Y. and J.O. Pedersen. 1997. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pp.412-420.

VITA

Name: Yue Wang

Born: Beijing, P. R. China

June 21, 1971

University attended: Beijing No.2 Foreign Language Institute

Beijing, P. R. China

Bachelor of Arts, 1990 - 1994