IMPROVING TEXT CLASSIFICATION PERFORMANCE WITH THE ANALYSIS OF LEXICAL DEPENDENCIES AND CLASS-BASED FEATURE SELECTION

by

Levent Özgür

B.S. in Computer Engineering, Boğaziçi University, 2001

M.S. in Computer Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in

Science and Engineering in partial fulfillment of

the requirements for the degree of

Doctor

of

Philosophy

Boğaziçi University

2010

IMPROVING TEXT CLASSIFICATION PERFORMANCE WITH THE ANALYSIS OF LEXICAL DEPENDENCIES AND CLASS-BASED FEATURE SELECTION

APPROVED BY:

DATE OF APPROVAL: 03.06.2010

ACKNOWLEDGEMENTS

My foremost acknowledgement goes to my supervisor Assoc. Prof. Tunga Güngör - without his support and patience in very critical milestones and his continuous guidance, this thesis would not be possible. I want to thank Prof. Fikret Gürgen and Assoc. Prof. Borahan Tümer for their continuous support, constructive criticism and feedbacks throughout all the thesis progress period. I would like to thank Dr. Suzan Üsküdarlı for her enthusiasm towards research ideas in general, and her comments and detailed corrections for my thesis report. I also would like to thank to Prof. Levent Akın for his valuable comments during the thesis defense and also his trust on me for offering a part-time instructorship position in the department during my studies.

I had a long and confusing journey for this thesis in the last seven years. Unfortunately, not being an idealist full-time PhD candidate researcher all the time, PhD dissertation seemed to be the main motivation that seriously affected all my critical life decisions in this period. During this long time, I had the chance and motivation to experience many related occupations in parallel - I worked full time in three companies (including a very tough entrepreneurship experience, Doğa Teknoloji), lectured in two different universities and participated in a research project besides my PhD studies. There are a serious number of valuable mates who somehow involved in some parts of this long journey - some of them even changed the way seriously. Physical limits of this section is surely not adequate to explicitly mention all of these valuable people. In general, I want to thank all my work-mates in the former companies (Lipman, Doğa Teknoloji and TIMw.e.) and universities (Boğaziçi University and Okan University), lab-mates (AILab), room-mates (ETA14) and more generally all CmpE-mates in the department, all my former and current house-mates in Akıncı Palace (!) and, of course, all my beloved buddies.

This thesis would not be completed in its way without the existence of the following core group from different associations - I would try to mention them with their most specific contributions during the journey: Arzucan Özgür (we, together, shaped the very critical introductive steps of the thesis), Murat Kayıhan (he was my beloved manager in Lipman during the lecture period of PhD, who permitted me very valuable times for PhD classes and research), Mehmet Gürmen (my beloved ex-partner who supported my PhD challenges during the proposal and early progress period in the impayable entrepreneurship experience: Doga Teknoloji and SeyYar), Prof. Ahmet Kaşlı (as the department head in Okan University, he showed great empathy, easiness and support to allocate very critical time for the thesis), Aslı Uyar Özkaya (most of the administrative and related decisions with also some critical technical issues in the long run of the thesis have a sign of her) and Prof. Nadir Yücel (his colorful life-worth experiences, priceless life advices and wisdom characteristics have a very strong impact on my thinking - I believe, he will completely recover from his recent illness and get well soon).

I gratefully acknowledge the financial support of Boğazici University Research Fund, number 05A103D and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.

Finally and most of all, I want to thank my parents Vicdan and Necati, my sister Demet with my beautiful niece İpek and my brother Murat. This thesis would be impossible to be finalized without their never ending love, encouragement, patience and support.

This thesis is dedicated to the memory of my father, Necati Özgür, who was also one of the pioneer textile engineers of Turkey. He was the first one to teach me the analytical way of thinking and showed the way to be a successful engineer and an overall fine and steady man. I am dreaming and feeling just as you are currently watching us, being happy and proud, Dad.

ABSTRACT

IMPROVING TEXT CLASSIFICATION PERFORMANCE WITH THE ANALYSIS OF LEXICAL DEPENDENCIES AND CLASS-BASED FEATURE SELECTION

In this thesis, we present a comprehensive analysis of the feature extraction and feature selection techniques for the text classification problem in order to achieve more successful results using much smaller feature vector sizes. For feature extraction, 36 different lexical dependencies are included and analyzed independently in the feature vector as an extension to the standard bag-of-words approach. Feature selection analysis is twofold. In the first stage, pruning implementation is analyzed and optimal pruning levels are extracted with respect to dataset properties and feature variations (words, dependencies, combination of the leading dependencies). In the second stage, we compare the performance of corpus-based and class-based approaches for feature selection coverage and then, extend pruning implementation by the optimized class-based feature selection. For the final and most advanced test, we serialize the optimal use of the leading dependencies for each experimented dataset with the two stage (corpus and class-based) feature selection approach.

For performance evaluation, we use the state-of-the-art measures for text classification problems: two different success score metrics and three different significance tests. With respect to these measures, the results reveal that for each extension in the methods, a corresponding significant improvement is obtained. The most advanced method combining the leading dependencies with optimal pruning levels and optimal number of class-based features mostly outperform the other methods in terms of success rates with reasonable feature sizes. To the best of our knowledge, this is the first study that makes such a detailed analysis on extracting individual dependencies and employing feature selection with two stage selection approach in text classification and more generally in text domain.

ÖZET

SÖZCÜKSEL BAĞIMLILIKLARIN VE SINIF BAZLI ÖZNİTELİK SEÇİMİNİN ANALİZİ İLE METİN SINIFLANDIRMA PERFORMANSINDA İYİLEŞTİRME

Bu tezde, metin sınıflandırma problemi için öznitelik çıkarımı ve öznitelik seçimi konuları üzerine çok yönlü çözümlemeler yapılmaktadır. Sınıflandırmanın çözümünde daha küçük boyutta öznitelik çözüm vektörü kullanarak daha başarılı sonuçlara ulaşmak hedeflenmiştir. Öznitelik çıkarımı konusunda, 36 değişik sözcüksel bağımlılık incelenmiş ve geleneksel sözcük-torbası dizisine en uygun halde eklenmiştir. Öznitelik seçimi ise iki aşamalıdır. İlk aşamada budama uygulaması yapılmış ve veri kümesi özelliklerine ve öznitelik çeşitlerine (sözcük, bağımlılık ve bağımlılık bileşenleri) göre en uygun budama düzeyleri bulunmuştur. İkinci adımda veri kümesi tabanlı ve sınıf tabanlı öznitelik seçimi yaklaşımları karşılaştırılmış; sonrasında budama işlemi, en başarılı olduğu tespit edilen sınıf tabanlı öznitelik seçimi ile geliştirilmiştir. Tezin son deneyinde; en başarılı bağımlılık tipleri, iki aşamalı öznitelik seçimi ile beraber kullanılmaktadır.

Başarı değerlendirmesi için, metin sınıflandırma problemlerinde kullanımı herkesçe kabul edilen iki ölçüm ve ek olarak üç değişik önemlilik testi uygulanmaktadır. Belirtilen değerlendirme ölçütlerine göre, önerilen her yeni yöntem, başarıyı önemli ölçüde arttırmaktadır. Bu duruma paralel olarak; sözcüksel bağımlılıkların en uygun kullanımını, iki aşamalı öznitelik seçiminin en başarılı düzeniyle birleştirdiğimiz son deney, genel olarak en başarılı sonucu vermektedir. Bu çalışma, bilgimiz dahilinde, metin sınıflandırmada ve genelde metin temelli problemlerde sözcüksel bağımlılıkların ve iki aşamalı öznitelik seçiminin çözümlemesi ve eniyilenmesi ile ilgili ilk detaylı çalışmadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

FN_i	False negatives of class i
FP_i	False positives of class i
$O(n^2)$	Space or time complexity representing quadratic complexity
TP_i	True positives of class i
π	Precision
ρ	Recall
acomp	Adjectival complement
adv	Adverbal clause modifier
agent	Agent
amod	Adjectival modifier
app	Appositional modifier
attr	Attributive complement
aux	Auxiliary passive
avg	Average
AW	All words
AW+	All words plus
AWDCKP	All words and optimal dependency combinations with keyword selection and pruning
AWDCP	All words and dependency combinations with pruning
AWDP	All words and dependencies with pruning
AWK	All words with keyword selection
AWKP	All words with keyword selection and pruning
AWP	All words with pruning
bow	Bag-of-words
cls	Clause modifier
comp	Complement
complm	Complementizer

conj	Conjunctive
Dmns	Domain based
dobj	Direct object
FtrNo	Feature number
Hyp	Hypernmy based
idf	Inverse document frequency
IN	Preposition
infmod	Infinitival modifier
iobj	Indirect object
LSI	Latent Semantic Indexing
macroF	Macro-averaged F-measure
microF	Micro-averaged F-measure
mark	Mark
MiniNg20	Mini 20 Newsgroups
nn	Noun compound modifier
NN	Singular noun
NNS	Plural noun
NP	Noun phrase
NSF	National Science Foundation Research Award Abstracts
obj	Object-verb
part	Participle modifier
PL	Pruning level
pobj	Prepositional object
prep	Prepositional modifier
prep-along	Prepositional modifier - along
prep-as	Prepositional modifier - as
prep-at	Prepositional modifier - at
prep-btwn	Prepositional modifier - between
prep-by	Prepositional modifier - by
prep-for	Prepositional modifier - for
prep-from	Prepositional modifier - from

prep-in	Prepositional modifier - in
prep-into	Prepositional modifier - into
prep-none	Prepositional modifier - generic
prep-of	Prepositional modifier - of
prep-on	Prepositional modifier - on
prep-over	Prepositional modifier - over
prep-to	Prepositional modifier - to
prep-with	Prepositional modifier - with
prt	Phrasal verb participle
POS	Part-of-speech
poss	Possession modifier
rcmod	Relative clause modifier
rel	Relative modifier
Rn	Ranked form
Reuters	Reuters-21578
S	Sentence
std	Standard deviation
stm	Stemmed form
subj	Subject-verb
SVM	Support Vector Machine
Syn	Synset based
synset	synonym set
tf	Term frequency
VP	Verb phrase
WND	WordNet Domains

1. INTRODUCTION

Classification can be defined as a process that tries to calculate a decision function based on training data with explicit input and output (classes) and later predicts the output of the new input set with minimum error using the estimated decision function [1]. Text Classification (TC), which is a sub-domain of classification and has been subject to active research for many years, is a learning task, where pre-defined category labels are assigned to documents based on the likelihood suggested by a training set of labeled documents.

Text classification is not an emerging problem in computer science, it is even one of the most historical and discussed topics in Artificial Intelligence. It is defined as *elephant among blind researchers* [2] which indicates its possible alternate views and several solution strategies. The solutions benefit directly from many other related disciplines in Computer Science (e.g. machine learning, natural language processing, Information Retrieval) or Mathematics (e.g. statistics, analytical geometry).

Text classification task may also be synonymously referred to as *text categorization*, *document classification*, *topic classification*, or *topic spotting* in the literature [3]. Automatic email sorting (or specifically filtering spam emails), sentiment detection of a text, assignment of news articles to topic channels, classification of the e-commerce customer logs / notes, detecting a document's encoding (ASCII, Unicode, UTF-8, etc.), or detecting the document language (English, Turkish, etc.) are all examples or sub-problems of text classification [2, 3]. Document management systems, search engines, semantic web applications, file organization, database organization / optimization are some examples of practical and widely used tools / concepts which are highly related with text classification by employing several common algorithms or approaches.

The main methods of TC domain problems are mostly supervised (with completely labeled training data) and parametric (fitting to a model and its parameters). In this sense, the bag-of-words (bow) form is accepted to be the simplest and the most successful approach to TC problems. In this approach, only the words in the documents are considered as features of the solution vector used for classification. It mostly relies on morphological concern by directly focusing on term frequencies and ignoring word order and possible relations between words.

The important issues and stages of text classification may be summarized as dataset selection, machine learning algorithms, preprocessing alternatives, feature representation, extracting new feature types (syntactic or semantic types), feature selection and performance measures. In this work, we focus on feature extraction and feature selection and delegate the remainder components to state-of-the-art approaches. Feature extraction extends the morphological concern of the bow approach by extracting new and informative features. The extracted features are mostly syntactic or semantic oriented which are expected to improve the classification performance by including some possible missing information of the standard approach. On the other hand, feature selection filters unimportant and uninformative features with particular statistical ranking rules in order to reach more scalable and accurate solutions.

For feature extraction, we make use of some syntactic features (e.g. part-of-speech (POS), lexical dependencies between words in a sentence) and some semantic features (e.g. synonym sets, concepts, etc.) retrieved from parsers and external knowledge bases, respectively.

For feature selection, there are various filtering methods (e.g. pruning, tf-idf, chi-square, information gain, etc.) which are analyzed and compared deeply in the literature [12, 58, 59]. In this work, we deal with the coverage approach of the metrics (class-based or corpus-based coverage of features) instead of another analysis of the selection methods which have been already discussed in many recent studies. The corpus-based approach uses a single feature vector for the discrimination of all the classes, while the class-based approach enables us to provide a distinct feature vector for each class. Pruning (filtering words according to their occurrence rates in the dataset) and selecting terms according to particular frequency formulas (i.e. tf-idf ranking [3]) of the features for each class are implemented for the basic and alternative selection metrics.

Our main goal is to develop a robust method in text classification that serializes the extraction of the most informative feature types with the optimal use of feature selection approaches.

1.1. Research Overview

The motivation of this thesis is to extend the standard bow approach used in the text classification problem with possible syntactic and semantic feature extraction by using optimal feature selection approaches. The main contributions are twofold.

On one hand, we performed an analysis for the optimal pruning of the features within 10 different levels. In the literature, usually an arbitrarily selected and small value (e.g. 2 or 3) has been used for this level. The analysis in this thesis was performed with respect to three main feature types (words, dependencies, and dependency combinations) and three datasets. Several optimal values were extracted according to these parameters (e.g. optimal level was analyzed as 13 for words and 8 for combination of dependencies in two datasets). We evaluated the incremental effect of 36 lexical dependencies in independent experiments with the optimal pruning levels and implemented the final experiment set-up by using the combination of the leading dependencies in addition to all the words in the documents. On the other hand, we compared the performance of corpus-based and class-based approaches for feature selection coverage and then, extended corpus-based pruning implementation by the tf-idf class-based feature selection. We examined a range between 10 and 4000 keywords and achieved the optimal parameters for the experimented datasets. For the final test, we serialized the optimal use of the leading dependencies for each experimented dataset with the two stage (corpus and class-based) feature selection approach.

Three significance tests have been implemented to test the robustness of the results and the significance of the improvements. Besides the classical micro and macro sign tests, we derived an extended version of the micro sign test for this study. The results show that for each extension in the methods, a corresponding significant improvement was observed in the success rates. The most advanced method, combining the leading dependencies with optimal pruning levels and optimal number of class-based features, mostly outperformed the other methods in terms of success rates with reasonable feature sizes.

The improvements achieved by the approaches were analyzed according to the algorithm details and a detailed study was performed to understand the reasons of these improvements with respect to the dataset properties. Significant patterns were extracted (e.g. formality level, skewness and text length) for the datasets which leads to the improvement rates with respect to the proposed algorithms (e.g. a massive improvement of macroF with a specific dataset was observed after the implementation of class-based feature selection due to the high skewness of the dataset). Another related analysis was performed with dependency use and it was shown that the formal datasets resulted in common and more complicated dependencies in the leading dependency analysis, while the informal dataset yielded other dependency types.

1.2. Thesis Outline

Chapter 2 presents a review of the state-of-the-art text classification fundamentals with an overview of the widely used supplementary methods and tools. Chapter 3 provides a detailed background of the current morphological, syntactic and semantic tools for the processing of the words and sentences in text classification.

Contributions of the thesis are presented mainly in Chapters 4-6. Chapter 4 presents the set-up details of the preliminary experiments with related tools and algorithms. Corresponding configuration settings are explained in order to analyze their practical discriminative power in the solution and find the possible most successful configuration pattern that will be informative and beneficial for text classification.

Based on the outcomes of the experiments presented in Chapter 4, Chapter 5 presents a comprehensive analysis of the lexical dependency and pruning concepts for the text classification problem. The pruning levels for words, dependencies, and dependency combinations for different datasets are analyzed. Statistically significant improvements are presented with the proposed approaches.

Chapter 6 focuses on feature selection approaches for text classification and introduces an extension of the optimum approach (presented in Chapter 5) with the analysis of class-based feature selection.

Although we provide a separate conclusion section for each experiment, Chapter 7 presents an overall conclusion and discussion of the contributions of this thesis.

2. FUNDAMENTALS OF TEXT CLASSIFICATION

In this chapter, text classification is discussed from several perspectives in all possible dimensions: datasets, solution alternatives, machine learning algorithms, preprocessing alternatives, feature properties (feature types, feature representation, feature selection, feature extraction) and performance measures. The set-up of all our experiments are introduced according to these fundamentals of text classification.

2.1. Datasets

We chose to work with English documents, since many well formatted and standardized English text classification datasets that contain training and test documents with predefined topics are available when compared to any other language.

Three well-known datasets from the UCI Machine Learning Repository [4] may be listed as: Reuters-21578 (Reuters), National Science Foundation Research Award Abstracts (NSF), and Mini 20 Newsgroups (MiniNg20). These datasets all include plain text without tagging or meta-data content - tagging is only used for the distinction and identification of each document topic and content in the dataset. These datasets have different characteristics, which may be critical for the classification performance. Skewness is one of the key properties of datasets that reflects the distribution of number of documents over classes. When a dataset's skewness rate is low, this means that it is a balanced dataset with approximately the same number of document samples per class. Allowance of multiple classes per document (indicating that documents in the dataset may belong to more than one topic), document length (short abstracts or long news articles), split (training and test sets) proportions, formality level (e.g. formal journals vs informal internet forum messages) are other properties of datasets.

Reuters is a well-known formal dataset that has been used in many TC algorithms [5, 6]. Standard Mod-Apte split partitions the dataset into 9603 training documents and 3299 test documents [5]. All the topics in both the training set and the test set were used in the experiments. The dataset, thus, consists of 90 classes, which is highly skewed. For instance, seven classes have only one document in the training set and most of the classes have less than ten documents. Also, the dataset allows multiple topics.

The NSF dataset consists of 129,000 abstracts describing NSF awards for basic research between the years 1990 and 2003 [4]. The level of formality of the dataset is high. It allows multiple topics. NSF is not a perfectly balanced dataset but its skewness rate is also not as high as Reuters. Document lengths are short due to its abstract content.

The MiniNg20 dataset consists of 2000 messages (split as 1600 for training and 400 for test), which is a collection of 100 messages from 20 different usenet newsgroups. Unlike the other two datasets, MiniNg20 is informal, with many grammatical errors, allows only one topic per text, and is a balanced dataset having equal number of messages for each topic.

2.2. Preprocessing Operations

Preprocessing is the initial step in dealing with the datasets that transforms the raw text into the required format for the feature representations of tools / algorithms for text classification. We may divide the preprocessing operations into two main parts: basic routines for feature formatting and more advanced routines with supplementary methods / tools for feature extraction.

2.2.1. Feature Formatting (Basic Routines)

These routines include the phases where documents are parsed, non-alphabetic characters and mark-up tags are discarded, case-folding is performed, stopwords are eliminated and morphological stemming is performed. Stopwords are some extremely common words that would appear to be of little value in classification of documents. Stoplists are used as the common groupings of stopwords for a particular problem.

Using a stoplist significantly reduces the size of the feature vector and the required system memory [3]. There is the standard list of 571 stopwords of the Smart system [7].

2.2.2. Feature Extraction (Advanced Routines)

Parsing (syntactic process where the POS information and dependencies of the sentences are extracted) and support of external knowledge bases for implicit features (semantic process which is covered by WordNet, Cyc, etc.) may be classified as advanced routines of preprocessing for text classification. Details about these processes and supplementary tools are explained in Chapter 3.

2.3. Document Representation

Bow form is accepted as the simplest and the most successful approach used in the TC problem. In this standard approach, only the words in the documents are considered as the features of the machine learning algorithm used for classification. Using a machine learning algorithm with this basic form is the straightforward, fundamental and conventional approach for text classification problems [3, 6]. In this approach, documents are represented by the widely-used vector-space model introduced by Salton et al. [7]. We represent each document vector as $\mathbf{d} = (w_1, w_2, ..., w_n)$ where w_i is the weight of the i^{th} term of document $\mathbf{d}$. Each dimension in the vector $\mathbf{d}$ stands for a distinct term (word) in the term space of the document collection based on the bow approach.

Representing the terms in this way causes the word ordering information within the sentences to be lost. String kernels with n-gram sequences, in which n consecutive words correspond to a single feature in the solution vector, feature were proposed to compensate for the ordering information and yielded promising results [8]. But this method has to deal with performance problems in large datasets - it suffers large space and time complexity issues and thus uses some approximation algorithms instead of representing the full structure. A different approach is making use of a language model (representing a document by the generation of new sentences from the document itself based on finite automata and probabilistic models) for text classification. Language models are sophisticated approaches used in information retrieval and they are considered as complicated for text classification [3]. These models are more appropriate for problems like query generation from texts, speech recognition, etc.

There are also recent studies that focus not only on the words, but, also the semantic concepts (synonyms, hypernyms, etc.) or syntactic features (POS, phrases, lexical dependencies, etc) in the solution vector by extending the bow approach. The syntactic and semantic feature types are extracted after applying the advanced preprocessing routines.

2.4. Term Weighting Approach

As stated in Section 2.3, the bow form is the conventional approach to the text classification problems in which documents are represented by the widely-used vector-space model. Term weighting approach is the key parameter for the stated feature representation.

As the common weighting approach, tf-idf metric is a simple and direct measure that takes the term frequency (tf) and the term's presence in the entire dataset (df) into account as shown in Equation 2.1. Boolean weighting checks only the occurrence of the term and does not consider the occurrence frequency so it is simpler than tf-idf but outperformed by tf-idf in related studies [5, 9]. Another alternative metric, Okapi BM25 is a non-binary model used mainly for query-document similarity measure, related search algorithms and relevance feedback [10].

$$tf\text{--}idf = tf_{t,d} * log\frac{N}{df_t} \tag{2.1}$$

We use the standard form of tf-idf [3] to calculate the weight of a term t in a document d, where tf is the frequency of term t in document d (each document vector is normalized to unit length to account for documents of different lengths), N is the total number of documents, and df is the number of documents in the dataset that include t.

2.5. Machine Learning Algorithm

In text classification, we have the document space X and there is a description of a document $d \in X$ and a set of classes $C = c_1, c_2, ..., c_m$, where m is the number of classes. By using machine learning algorithms, we try to learn a classification function f that maps documents to classes [3]:

$$f : X \rightarrow C \tag{2.2}$$

The main machine learning approaches used in the TC domain may be classified as supervised (e.g. Support Vector Machine (SVM)) vs semi-supervised (e.g. Naive Bayes with Expectation Maximization), parametric (e.g. SVM, Naive Bayes) vs. non-parametric (e.g. k nearest neighbor) methods, linear (e.g. SVM with linear kernel) vs. non-linear (e.g. SVM with radial basis kernel) classifiers, vector space (e.g. SVM, artificial neural network, Rocchio) vs. probabilistic (e.g. Naive Bayes) classification, and decision tree modeling (e.g. rule-based decision trees). Clustering (e.g. k-means, which is unsupervised and semi-parametric) may also be employed in the case of the existence of a dataset without labeled training data.

SVM is a technique introduced in 1995 by Vapnik, which is based on the Structural Risk Minimization principle [13]. It is mainly designed for solving two-class pattern recognition problems. Figure 2.1 represents the members of the two classes with positive $(+)$ and negative $(-)$ signs. The main motivation is to find the decision surface (hyperplane) that separates the positive and negative training examples of a class with maximum margin as seen in Figure 2.1 in order to decrease the classification errors with the test data. The dashed lines which are parallel to the main line show how much the decision surface can be moved without resulting in classification errors. Margin is the distance between these parallel lines. Support vectors, which are marked with circles, are the members of the two classes which are closest to the decision surface.

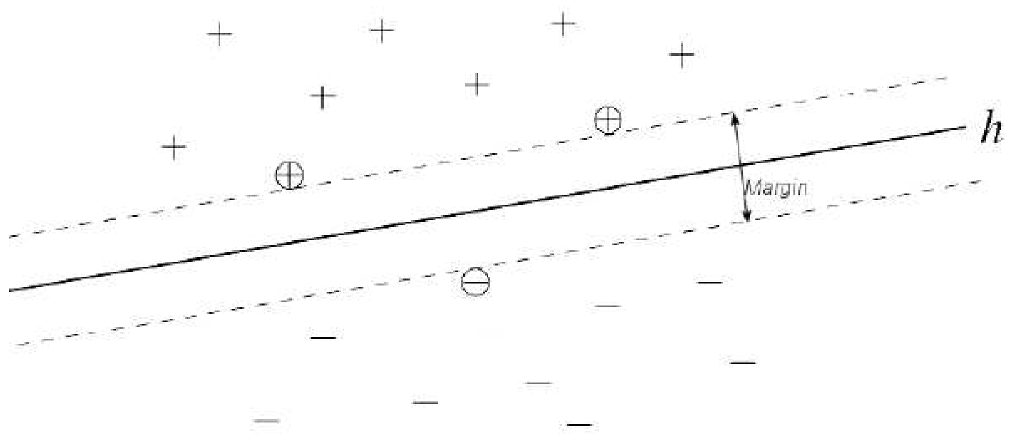


Figure 2.1. SVM fundamentals: support vectors, hyperplane and margin

Several studies have compared the performances of these approaches and in general, SVM with linear kernel was shown to yield the best results [5, 6, 11, 12]. For the fundamental challenges in the text classification domain (high dimensionality, sparse instances, separability of classes), SVM provides efficient solutions by being more immune to the overfitting problem, using an incremental algorithm with an inductive bias that suits problems with dense concepts and sparse instances, and employing a basic linear separation model that fits the discrimination of most of the classes [11]. In the initial tests, we also experimented neural networks with multiple layers as an alternative approach for the machine learning part, but the method caused much more time complexity without yielding more successful results when compared with SVM.

2.6. Feature Selection

The basic idea of the feature selection implementation is to select the most discriminative ones among all the features for the classification problem. More successful scores with less (but more informative) feature numbers is aimed with this selection process.

Pruning is a simple and efficient approach to reach this smaller but more discriminative feature set. The pruning process basically focuses on the total frequency of the features in the whole dataset and filters out less frequent ones. This is generally performed with predefined static levels (e.g. words occurring less than 2 or 3 times are eliminated) without an optimization study.

Other metrics (mutual information, chi square selection, bi-normal separation, tf-idf, etc.) may also be used for term selection. These metrics involve mostly arithmetic calculations of true / false positives and true / false negatives with their counts, rates or percentages. They usually differ with their decision criteria with respect to the recall and precision metrics. Tf-idf is one of the most widely used algorithms in this scope [3]. Tf-idf is basically the multiplication of *tf* and *idf* values that has been used for term weighting approach. This metric focuses on the frequency of the term in a document with a discount of its general occurrence in the whole dataset.

Latent Semantic Indexing (LSI) is another type of feature reduction approach but differs from the other mentioned ones with its semantic perspective - the method represents the documents and terms in the same space by allowing the underlying semantic relationships between them [14]. Co-occurrence of data is important in LSI, the words that occur together in the same document are said to be semantically related and similar. This method has been stated as not satisfactory enough when applied directly to the whole training dataset, instead local LSI methods have been analyzed to improve classification performance [15].

2.7. Performance Measures

Performance measures are needed in classification problems to evaluate the success of each of the proposed algorithm (success scores) and provide a mechanism for statistical testing to compare the success scores of these algorithms (significance measures).

2.7.1. Success Scores

To evaluate the performance of the proposed approaches, we use the commonly used F-measure metric, which is equal to the harmonic mean of precision (π) and recall

(ρ) [3]. π and ρ are formulated as follows:

$$\pi_i = \frac{TP_i}{TP_i + FP_i}, \quad \rho_i = \frac{TP_i}{TP_i + FN_i} \tag{2.3}$$

Here, TP_i (True Positives) is the number of documents assigned correctly to class i; FP_i (False Positives) is the number of documents that do not belong to class i, but are assigned to this class incorrectly; FN_i (False Negatives) is the number of documents that are not assigned to class i by the classifier but which actually belong to this class.

The F-measure values are in the interval (0,1) and larger F-measure values correspond to higher classification quality. The overall F-measure score of the entire classification problem can be computed by two different types of average, micro-average and macro-average [3].

In micro-averaging, F-measure is computed globally over all category decisions:

$$F(micro\text{–}averaged) = \frac{2 * \pi * \rho}{\pi + \rho} \tag{2.4}$$

Micro-averaged F-measure (microF) gives equal weight to each document and is therefore considered as an average over all the document/category pairs. It tends to be dominated by the classifier's performance on common categories.

In macro-averaging, F-measure is computed locally over each category i first and then the average over all categories is taken:

$$F_i = \frac{2 * \pi_i * \rho_i}{\pi_i + \rho_i}, \quad F(macro\text{–}averaged) = \frac{\sum_{i=1}^{M} F_i}{M} \tag{2.5}$$

where M is total number of categories. Macro-averaged F-measure (macroF) gives equal weight to each category, regardless of its frequency. It is influenced more by the classifier's performance on rare categories. We provide both measurement scores to be more informative.

2.7.2. Significance Measures

Sign test is the common algorithm in TC related problems to understand the significance of the improvements of the proposed approaches [6]. Similar to success scores, two types are employed:

- Micro sign test: This is an instance-based test that compares the system based on the micro perspective of the results. A document-category pair is the basic unit to decide whether the document belongs to the category (positive instance - 1) or not (negative instance - 0). In this significance test, two systems are compared based on their binary decisions on all the document-category pairs. The correctness of the decisions are compared for each pair [6]. Standard z values are calculated and the corresponding confidence levels are determined according to the standard normal distribution [16, 17].
- Macro sign test: This is a category-based test that compares the two systems based on their F-scores on each category of the dataset. The test considers the number of times that the two systems yield different scores and the number of times that the score of one of the systems is larger than the score of the other system [6]. The z value calculation and confidence level determination processes are the same as in the micro sign test.
- Micro sign test with positive instances: This a variance of the Micro sign test that we have suggested in order to consider only positive document-category matches for significance calculation. The document-category matrix in the micro sign test is a highly sparse matrix with a large number of negative instances (0). This is not surprising because each document belongs to usually three or four categories at most and has a negative value for the remaining categories. The micro sign test takes all the positive and negative instances between the compared systems into consideration, which in fact favors the negative ones since they occur much more frequently. Based on this observation, we decided to derive an extension of the micro sign test by redesigning the test considering only the positive instances. In this way, the test focuses only on the instances in which the document belongs to that category. The rest of the test (comparison algorithm,

z value calculation, confidence level determination, etc.) is the same as in the micro sign test. In specific situations that we want to consider only the positive matches of document-category matrix for performance comparison, the outcome of this extended version of micro sign test should be analyzed.

2.7.3. Other Measures

Space complexity and time complexity are also two other main properties of a proposed computational system. Space complexity provides information about the amount of storage space required by an algorithm, this measure is parameterized with the size of the problem. For example, if the space complexity is mentioned as $O(n^2)$, the required space expands to four times of its previous value in case the problem size doubles. On the other hand, time complexity measures the number of steps (time) required by an algorithm varying with the size of the problem. Its symbology is the same as the space complexity. $O(n^2)$ indicates the expansion of required steps in terms of time domain with respect to problem size.

3. SUPPLEMENTARY TOOLS FOR PREPROCESSING IN TC

After discussing the main stages of TC in the previous chapter, we will now analyze the necessary supplementary tools for the problem. This chapter provides the details of the current morphological (e.g. stemmer that finds the root form of the words), syntactic (e.g. parser that extracts the POS information and dependencies of the sentences) and semantic (e.g. ontology that extracts implicit concepts of the words) tools for the processing of the words and sentences in text classification.

3.1. Morphological Knowledge

Besides the basic preprocessing routines of the words in the documents (discarding non-alphabetic characters and mark-up tags, case-folding, and stopwords elimination), stemmer analysis in text processing is the main perspective of the morphological concern with text classification. From the classical view of morphological stemming, implementation is based on only morphological issues that are completely independent from the syntactic and semantic structure of the sentence.

Porter stemmer is the most common algorithm for stemming in English [3]. The algorithm consists of five phases of word reductions which are applied sequentially. Within each phase, there are various conventions to select rules. Table 3.1 lists an example set of rules and the corresponding samples for the conventions used in the first phase [3].

An alternative to stemming is the *lemmatizer* concept, which performs syntactic analysis in addition to morphological analysis. In this process, stemming is performed by removing inflections of the word but conserving the derivational affixes based on the syntactic analysis of the sentence. We utilize both ways of stemming by employing Porter Stemmer as the morphological stemmer and Stanford Lemmatizer, the built-in stemmer of Stanford Parser, as the alternative one.

Rule		**Example**	
Before	**After**	**Before**	**After**
sses	ss	caresses	caress
ies	i	ponies	poni
ss	ss	caress	caress
s		cats	cat

Table 3.1. Stemming rules with their examples

3.2. Syntactic Knowledge

This section provides details about the lexical dependency concept which is a type of document pattern. The Properties of the Stanford Parser, dependency extraction with this parser and the use of dependencies in Information Retrieval problems (including text classification related references) will be explained.

3.2.1. Document Patterns and Lexical Dependencies

Document patterns represent domain specific information with a specific structure (pattern) which are extracted from unstructured machine readable documents [18]. Different dependency types (predicate-argument model, chains, linked chains, subtrees, etc.) have been studied in the literature using alternative approaches to linguistic analysis [19]. Building a model based on document patterns basically aims to extract sufficiently expressive patterns from the documents without introducing much complexity.

Lexical dependency is an extended model of document pattern in which sentence structure is represented using the grammatical relations (object-verb, conjunctive, prepositional modifier, etc.) between the words in a sentence [20]. A dependency is simply formed as the combination of any two words with any of these grammatical relations. For example, three sample dependencies extracted from a sample sentence

"*We use combination of dependencies in text classification.*" may be listed as *we-use* (subject-verb), *classification-text* (noun compound modifier), and *dependencies-classification* (prepositional modifier). Here, *we* is the subject and *use* is the main verb of the sentence and the combined dependency forms the subject-verb dependency. *Text* and *classification* are both nouns and the preceding one affects the meaning of the other, so this dependency couple is named as noun compound modifier. For the last dependency, *dependency* is modified by the noun *classification* through the *in* preposition which forms the prepositional modifier dependency.

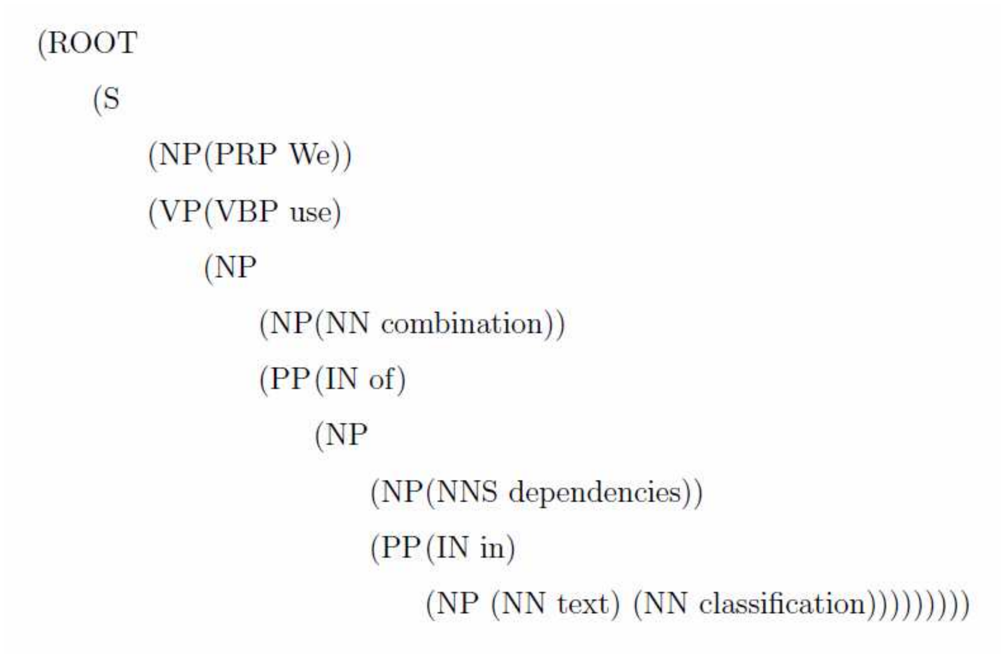
3.2.2. Properties of Stanford Parser

Stanford Parser is known to be one of the most powerful and efficient parsers having the least error rate [19]. Given a sentence, the parser identifies the dependencies in the sentence in two phases. In the first phase, the sentence is parsed using a statistical phrase structure parser based on a probabilistic context-free grammar, which was trained on the Penn Wall Street Journal treebank [21]. The POS tags of the tokens, the semantic heads in the sentence, and the dependents of the heads (auxiliaries, complement, etc.) are determined. In Table 3.1, we show the parse tree of the example sentence "*We use combination of dependencies in text classification.*".

Here, S stands for the main sentence, NP is noun phrase while VP is verb phrase. Other abbreviations stand for the POS information mainly. PRP is proper noun, VBP is verb with present tense, NN is singular noun, IN is preposition and NNS is plural noun.

In the second phase, the dependencies extracted are labeled with grammatical relations by using the tree-expression syntax defined by the tregex tool [22]. We list below the dependencies obtained for the example sentence (details of the dependencies will be explained in Section 5.2.2):

- subject-verb (*We, use*)
- object-verb (*combination, use*)



```
(ROOT
    (S
        (NP(PRP We))
        (VP(VBP use)
            (NP
                (NP(NN combination))
                (PP(IN of)
                    (NP
                        (NP(NNS dependencies))
                        (PP(IN in)
                            (NP (NN text) (NN classification))))))))
```

Figure 3.1. Sample parse tree

- prepositional modifier - of (*combination, dependencies*)
- noun compound modifier (*classification, text*)
- prepositional modifier - in (*dependencies, classification*)

In our tests with the Stanford Parser, we observed that the parser averts syntactic ambiguities in the sentences successfully and gives the first probable parse as the result.

3.2.3. Usage of Syntactic Information in Text Related Problems

A study in the literature focused on syntactic tools for text categorization and showed some advantages of the context-sensitive text categorization methods [23]. POS information has been utilized in another study in which only the type of the POS information is used (not the content) [24], which has been analyzed to be surprisingly effective for text classification.

Another POS utilized algorithm considers only specific POS information for feature selection criteria (e.g. only nouns, verbs or proper nouns are selected as keywords) [25]. Their results suggest that relevance of POS information may be a strong indicator for the preprocessing phase of feature selection in text classification.

The concept of lexical dependency was previously used in many information retrieval applications such as sentiment analysis [26], parse disambiguation [27], machine translation [28], textual entailment [29] and discourse coherence [30]. It was also employed as a common framework for interactive, multimodal, and multilingual information retrieval problems that also included text classification implementation [31].

There are studies that specifically focus on dependencies for text classification. The pioneering studies in this topic include noun phrases and main argument dependencies (subject-verb, object-verb, etc.) in text classification, however, no significant improvement was achieved [32, 33]. In a more recent study, dependencies (extracted by n-gram rules) were used in the solution vector in addition to words and significantly more successful results were yielded, however, only the leading dependencies were used and the selection process required human interaction, which was performed by human annotators [34]. In another recent study, many linguistic features (e.g. POS information, complex nominals, proper nouns, and word senses) were experimented in addition to the words, but no significant improvements were observed [35]. Another study [36] reported that linguistic processing did not improve the bow approach and also pointed out the negative effect of a specific dependency: subject-object-verb. In a related study which used dependencies by capturing frequently occurring keyword combinations within short segments (rule-based algorithms), reported successful results with a specific and not widely used dataset [37]. Another recent study increased the success rates of the classifier by combining the bow approach with a necessary noun-modifier dependencies and word senses [38].

In almost all of these studies, dependencies were altogether included in the solution vector without any further analysis. Another drawback regards pruning: It was mostly performed during tests but with a predefined static level (e.g. two or three). In a very recent study which performed a distinct analysis of dependencies, a slight improvement over the baseline of the standard bow approach was achieved [39]. On the other hand, due to the lack of pruning, most of the dependency types used in [39] yielded many instances (distinct word pairs), which resulted in an excessive number of features and a highly sparse solution set in the machine learning algorithm.

3.3. External Knowledge Bases - Ontologies

Ontology refers to the shared understanding of some domains of interest. It is often conceived as a set of classes (concepts), relations, functions, axioms and instances [40]. It is defined as a formal, explicit specification of a shared conceptualization. *Conceptualization* refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. *Explicit* means that the type of concepts used and the constraints on their use are explicitly defined. *Formal* refers to the fact that the ontology should be machine readable. *Shared* implies that an ontology should capture consensual knowledge accepted by the communities [40].

WordNet and Cyc are accepted as the most general ontologies with extensive content. In this part of the report, we focus on these ontologies.

3.3.1. WordNet

WordNet is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets (synsets), each representing one underlying lexical concept.

Different relationships interrelate the WordNet synsets. The well-known and most useful ones are the *hyponymy / hypernymy* and *meronymy / holonymy* relation couples.

The *hyponymy / hypernymy* is the main relation group of the WordNet hierarchy which defines *isA* relationship between concepts. For example, student is a hyponym (child) relation of the person synset and a person is a hypernymy (or parent) of student. Both relations are transitive and asymmetrical. Each concept has a single parent, except the root concept which is the hypernmy of all the other concepts and is semantically empty.

Another relation couple is the *meronymy / holonymy* which defines the *has part / part of* relationships, respectively, between synsets. For example, wheel is a *holonymy* of car synset and car is a *meronymy* of wheel synset.

As for the drawbacks of Wordnet, it is said to be an ontology heavily based on the lexicon concept and a hierarchy without any further analysis on the syntactic features and semantic concepts of the whole sentence. Also, WordNet has a limited number of private names / instances (people, organizations, geographic locations, books, songs, etc.) of the concepts.

3.3.2. WordNet Related Tools

There are some recent supplementary tools utilized with WordNet. One of them is WordNet Domains (WND). In WND, there are about 200 uniquely defined domains. Each synset in WordNet is annotated with some domain(s)[41, 42].

Another supplementary tool is WordNet Affect which is in fact an extension of WND [45]. A linguistic resource has been presented for the lexical representation of the affective knowledge. WordNet Similarity tool is another supplementary tool of Wordnet. This tool mainly computes a meaning / similarity distance between two WordNet synsets.

3.3.3. Cyc Ontology

Cyc ontology is accepted to be the most general ontology in the world [46]. It is a formalized representation of a vast quantity of fundamental human knowledge: facts, rules of thumb, and heuristics for reasoning about the objects and events of everyday life. The medium of representation is the formal language CycL [47]. CycL is a large and extraordinarily flexible knowledge representation language. It is essentially an augmentation of first-order predicate calculus, with extensions to handle equality, default reasoning and some second-order features. OpenCyc is the open source version of the Cyc system and ResearchCyc is the extended version of OpenCyc with a license granted for academic research.

ResearchCyc supports several user interfaces for skill levels ranging from expert to novice. The Cyc Browser consists of dynamic (CGI-generated) HTML pages that allow experienced users to query, browse, edit, and add contents to the knowledge base. The Cyc Fact Editor is a template-based Java interface that allows users to add, view, edit facts (ground assertions) to the knowledge base. The Query Library is a Java interface that allows users to pose pre-formulated queries (i.e. libraries of queries already defined for particular domains), and to compose new, arbitrarily complex queries by assembling and modifying pre-existing query fragments and templates.

From the perspective of implementation detail, Cyc holds the *isA* relation (e.g. Bill Clinton is a (PastOrPresentPresidentFn UnitedStatesOfAmerica) HumanAdult Individual MaleHuman LeftHandedHuman FamousHuman UnitedStatesPresident). The membership relation is also held (e.g. Dog is a member of (CanisGenus CanineAnimal DomesticatedAnimal). Query implementation is possible through several interfaces (e.g. web and application programming interface) in Cyc (e.g. isQueryTrue : (LIST likesAsFriend BillClinton JimmyCarter) PeopleDataMt). Relative concepts may be listed or queried (e.g. Ankara and Athens are both capitals). We may modify the internal data with specific functions such as *assertGaf* and *unassertGaf*. Concept extraction for a given word or word group is stated to be possible with functions: *sget-DenotsOfString-getMWSDenotsOfString*. Cyc is also stated to support lexical analysis and give related concepts for the words in the sentence with the specific function *denotationMapper*. However, since the analysis is performed word by word, it does not yield an optimal time complexity and does not consider the structure of the sentence.

3.3.4. Other Ontologies

As an example of other general-purpose ontology variants, Wikipedia is a human-readable and global knowledge base that contains millions of articles collaboratively written by volunteers [48]. However, Wikipedia lacks one of the fundamentals of ontology standards: the concepts and hierarchical relationships among them are not formally defined.

Another general-purpose ontology is Open Mind Common Sense, which aims to build and utilize a large common sense knowledge base formed by many global volunteers [49]. ConceptNet is the semantic engine for this ontology [50]. ConceptNet was employed to retrieve images with text descriptions by focusing on spatial relationships. The evaluation on test data of the 2005 ImageCLEF showed that integrating commonsense knowledge in information retrieval was feasible [51].

3.3.5. The Use of Knowledge Bases in Text Classification

WordNet is the most common and popular lexical tool utilized in text classification studies [52]. Most of the studies focuses mainly on synsets for text classification and expands the keyword vector with synsets of the words. One of the studies has stated an increase in the overall success but synset disambiguation has been performed manually instead of a successful and ideal sense disambiguation [53]. There is also a recent study focusing on WordNet features [54]. They have implemented a text classifier shell that has incorporated various WordNet features into the standard category model and they have reported no effective success increase with their detailed tests.

In one of the recent studies, WND has been utilized for domain acquisition [43]. The purpose was to identify how the global conceptual representation of a sentence could contribute to the resolution of the classification problem. For this purpose, they presented an approach where the meaning of a sentence has been represented with its associated relevant domains. Another domain-specific study covers domain-specific sense extraction [44].

4. PRELIMINARY TESTS

The main motivation of this chapter is to recognize and experiment with several tools and algorithms using different configuration settings in order to analyze their practical discriminative power and discover the possible most successful configuration pattern which is informative and beneficial for text classification. The configuration with the optimal performance is extensively analyzed and improved in the following chapters.

4.1. Common Preferences of the Experiments

We have provided the main definitions and properties of the fundamentals of text classification in Chapter 2. In this section, we discuss and explain our main preferences (e.g. algorithms, tools, etc.) for the discussed topics. We will use these preferences in the following experiments in this thesis, so the below given discussion may be considered as a structural outline of the experiments.

4.1.1. Datasets

Preliminary experiments will be performed mainly with Reuters, and partially with NSF datasets. For the discriminative experiments of the proposed approaches, the experiments will be performed in all the three datasets from the UCI machine learning repository: Reuters, NSF and MiniNg20 [4].

We perform the standard use and split of Reuters and MiniNg20 datasets. We randomly selected the year 2001 data from NSF and used five sections (four sections for training and one section for test). We form five different splits, repeat all the tests with these five cross folds, and take their average as the final result.

4.1.2. Preprocessing Operations

We analyze the preprocessing operations in two main parts: basic routines for feature formatting and more advanced routines with supplementary methods / tools for feature extraction.

Preprocessing for feature formatting is the initial and basic step of text processing where documents are parsed, non-alphabetic characters and mark-up tags are discarded, case-folding is performed, and stopwords (for word features) are eliminated. We use the list of 571 stopwords of the Smart system [7]. Using a stoplist significantly reduces the feature vector size and the memory requirements of the system [3]. In our initial tests where stopwords were not eliminated in extracting the word features, we observed a 2%-10% (depending on the dataset and the pruning level) increase in the size of the solution vector with no significant change in the success rates. On the other hand, when used with phrases and dependencies, it was stated that stopwords lead to a more effective and precise analysis [3]. So we did not use stoplist filtering during dependency extraction, which led to dependencies including stopwords as well (e.g. *write down* - a phrasal verb participle dependency). For stemming, after analyzing several alternatives, we chose the Porter stemmer which is one of the most experienced stemmers for word forms [39, 55].

For feature extraction, POS and dependency extraction by the parser and the support of external knowledge base (WordNet, WND, Cyc, etc.) is alternatively used with respect to the properties of the experiments.

4.1.3. Document Representation

Documents will be generally represented as a feature vector in accordance with the classical learning algorithms. The bow approach will be employed as the benchmark in most of the algorithms (exceptions are the experiments that use only concepts or dependencies without words) and will be extended by dependencies, POS information, synsets, domains, etc. according to the experiment properties.

4.1.4. Term Weighting Approach

Our main motivation in this study is not to analyze the feature representation methods for text classification but instead, utilize the most efficient and simple method to measure the improvement of the proposed approaches. So we have selected the commonly used tf-idf for our study and tried to optimize its use in the experimented algorithms. For the optimized tf-idf calculation, each document vector is normalized so that it is of unit length to account for documents of different lengths [5].

4.1.5. Machine Learning Algorithm

SVM with linear kernel is the machine learning module we use as the classification algorithm in all the experiments.

4.1.6. Feature Selection

Our main motivation in this study is to extract the most successful features for the TC problem and use them optimally in order to benefit from them in the most efficient way.

Preliminary experiments are performed using all possible features without a selection process. For the advanced experiments, the basic feature selection algorithm (pruning) is implemented based on the term's frequency in the whole dataset. Optimization of the pruning levels with different feature types (word, dependency) and different datasets are performed.

In the initial tests, we also used tf-idf as an alternative method for feature selection, which is one of the most widely used feature selection metrics [3]. We obtained similar success rates as the pruning implementation when only the words were used, but the success decreased when the dependencies were included in the feature vector. Thus, we decided to continue with the pruning technique. Using a feature selection metric on dependencies may necessitate a detailed analysis and we leave the study of combining possible feature selection metrics with pruning for both word and dependency features to Chapter 6 as an independent analysis.

4.1.7. Performance Measures

All the experiments are evaluated with microF and macroF scores. For the discriminative experiments of the proposed approaches, the algorithms will be statistically compared with micro sign test, macro sign test and micro sign test with positive instances in order to measure the significance of the improvement.

For the significance tests, we use the following symbols and terminology to denote the result of a comparison:

- $\gg$: significantly outperform (more successful with at least 99% confidence level)
- $>$: significantly better (more successful within 95% - 99% confidence level)
- $\sim$: similar - not significantly better or worse (success confidence level is less than 95% and more than 5%)
- $<$: significantly worse (less successful within 95% - 99% confidence level)
- $\ll$: significantly defeated (less successful with at least 99% confidence level)

Space and time complexities are two important performance measures which are also very critical for some related and specific problems in the literature (query optimization in a search engine or a data warehouse). But these measures are not the main concern for the text classification problem within the predefined datasets, hence we will not explicitly analyze the corresponding complexities in detail.

For a brief outline, we will provide the general situation for the implementations. All experiments are implemented in Hp Workstation xw6200 with Xeon CPU 3.2 GHz and 4 GByte RAM.

The most time consuming parts of the experiments involve access to external resources (WordNet, Stanford Parser, etc.). Dataset parsing is the most time consuming part of the overall process. Reuters was parsed in 55 hours while the same operation took 15 hours for NSF. However, this parsing operation is performed offline and only once to be utilized for all the test modes for that dataset. For the WordNet support, the preprocessing phase in the training phase is another relatively time consuming part which takes about approximately 15 minutes for only pattern synset extraction and about 90 minutes for the synset extraction of all words within approximately 10000 documents.

For the proposed algorithms, the most time consuming part is the creation of tf-idf values for all existing terms in the training and test phases. This process is analyzed as $O(n^2)$ - it takes approximately 50 minutes with about 20,000 keywords.

4.2. Implementation Details of the Related Tools in the Experiments

This section discusses the implementation details of the related tools / methods (SVM, Cyc, Stanford Parser, WordNet and WND) that will be used commonly or optionally in the experiments. As seen in Figure 4.1, a brief system architecture covering the generic parts of the system is presented. The proposed system implements the preprocessing of the dataset documents, uses external knowledge bases (e.g. WordNet) for conceptual features, parses the sentences with Stanford Parser and finally performs classification of the documents by using SVM. Further details with specific use and parameters of the tools / methods will be explained in the related experiment sections.

4.2.1. Parser Implementation Details

We use Stanford Parser tool (version 1.5) as the parsing module, which is known to be one of the most powerful and efficient parsers having the least error rate [19]. Two main submodules are implemented with Stanford Parser as seen in Figure 4.2 :

PreParse: Dataset documents are preprocessed to obey the Stanford Parser input file regulations - mostly the headers and taggers are regulated in order not to interfere with the sentence parse structure.

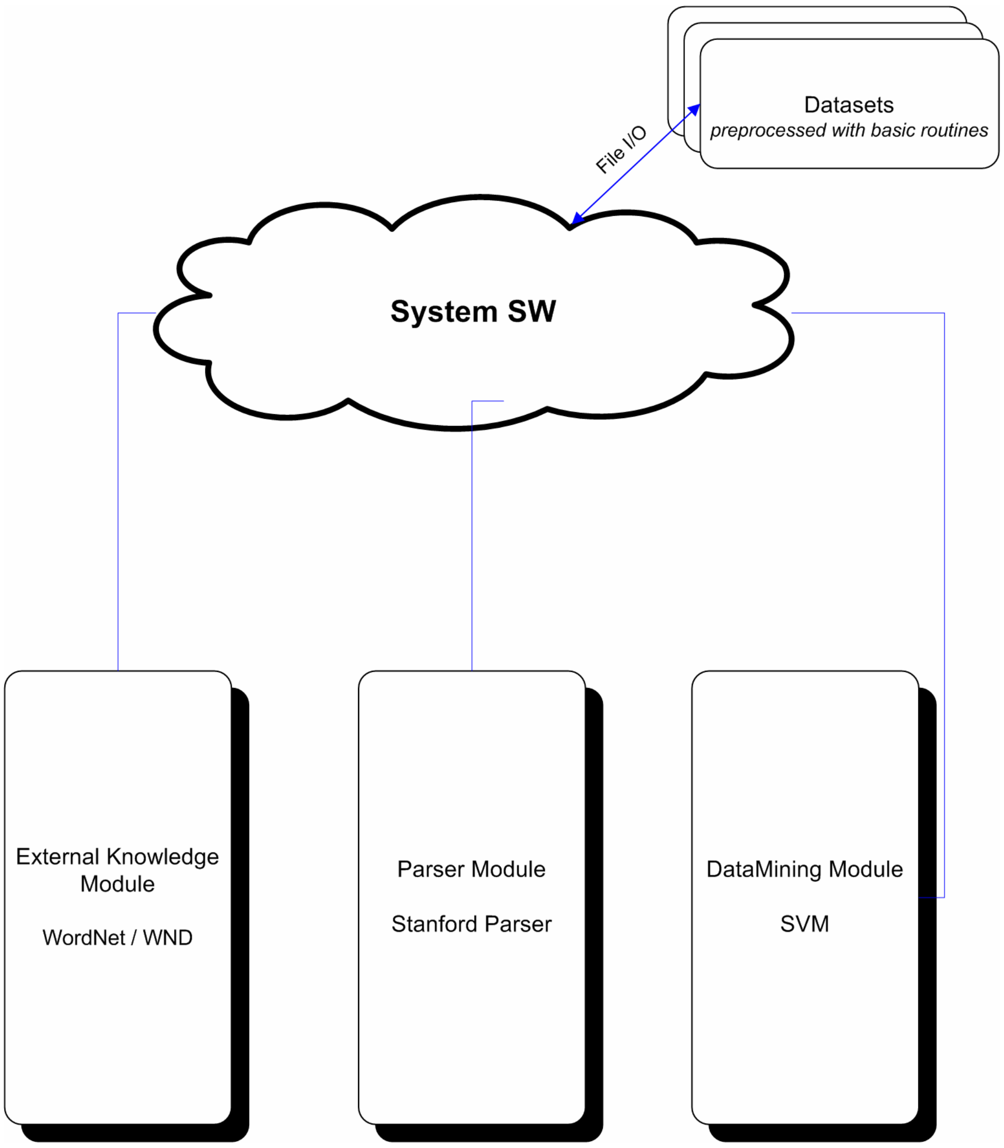


Figure 4.1. General system architecture

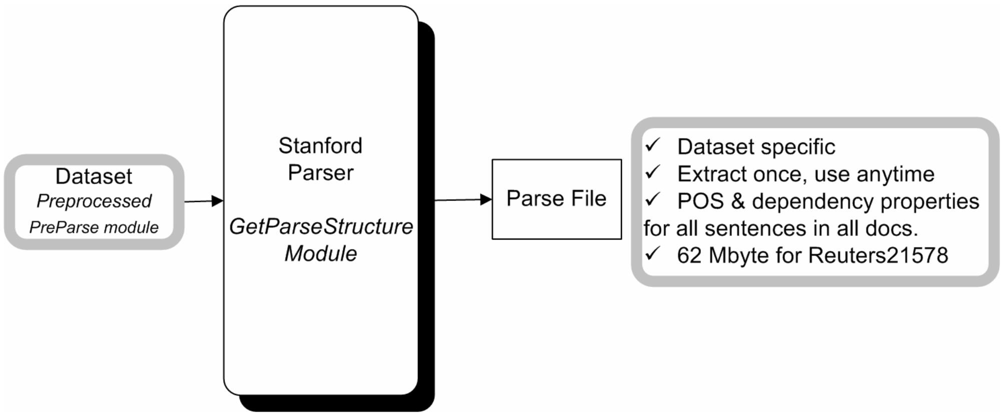

Dataset
Preprocessed
PreParse module
Stanford Parser
GetParseStructure Module
Parse File
✓ Dataset specific
✓ Extract once, use anytime
✓ POS & dependency properties for all sentences in all docs.
✓ 62 Mbyte for Reuters21578

Figure 4.2. Stanford Parser usage and parse file

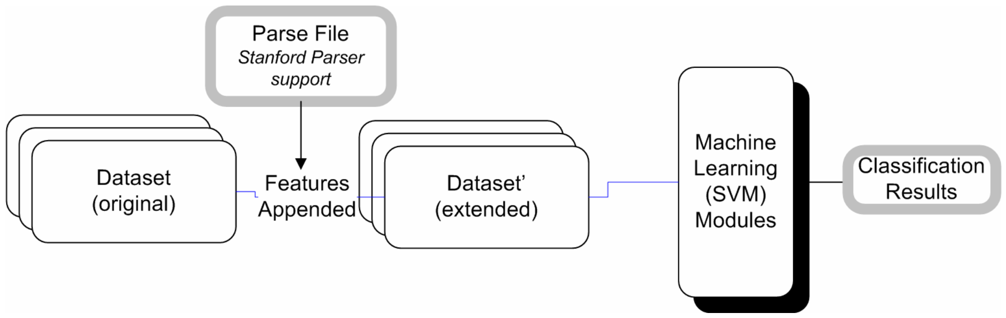


Figure 4.3. Usage of parse file

GetParseStucture: Two main features types are extracted from each sentence: POS (noun, verb, etc.) and dependency (object-verb, prepositional modifier, etc.) property. All the related features for all the sentences in each dataset are processed once and kept in parse file for any further use (e.g. all possible classification tests). Thus, the extracted parse file is dataset specific and used in the system by the machine learning module as in Figure 4.3. Size of the parse file is quite large when compared with the original dataset sizes: Reuters parse file is about 62 MByte while NSF and MiniNg20 are about 15 Mbyte.

4.2.2. SVM Implementation Details

We use the SVM^{light} system which is an efficient implementation by Joachims [11] and has been commonly used in previous studies. Three main submodules are implemented with SVM:

PreSVM: Input files are standardized in accordance with SVM style.

BatchSVM: Batch SVM program is executed for all classes. Class comparisons are performed by the one-versus-all style in which each class is compared against the aggregation of all the possible classes for the assignment of each document so the batch program is called for each different class [56].

PostSVM: SVM output data is formatted for success scores (microF and macroF).

4.2.3. Cyc Implementation Details

The last release of Cyc within ResearchCyc project (Release 1.1) claims to perform POS disambiguation in addition to lexical study with sentence-level syntactic analysis with an extended semantic analysis. On the other hand, the current implementation of Cyc was not yet satisfactory when compared with the stated properties. Cyc handles membership, definitions and related concepts at lexicon level but it lacks critical aspects such as syntactic and semantic analysis at sentence level. In addition, our implementation attempts suffered from large time complexities (even halted in some cases) of the function calls. Therefore, we did not continue the Cyc support in the system.

4.2.4. WordNet and WND Implementation Details

We use the online lexical reference system WordNet 2.0 as the external knowledge resource and its related tool WND to extract more general information about the words. The basic use of WordNet and additional WND support is implemented with the following steps:

(i) Stem each word in the corpus with the selected stemmer / lemmatizer
(ii) Find POS information for the word with Stanford Parser support
(iii) Find the corresponding synset id by querying WordNet with the stemmed word and its POS information. If synonmy option is selected, return the corresponding synset id. Else if hypernymy option is selected, return the corresponding synset id of the hypernymy (parent) of the synset by querying WordNet with the synset id. Else if WND option is selected, return the corresponding domain by querying WND with the WordNet synset id.

4.3. Introductory Experiment - Standard Bow Approach in TC

For the initial experiment set-up of the proposed system, the details of the benchmark method is the standard use of bow approach with all words (AW) for TC.

4.3.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

4.3.1.1. Datasets. Reuters is used as the main dataset for the experiments.

4.3.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Porter is used for stemming implementations.
- Feature Extraction: Experiments are performed without parser and external knowledge base (WordNet, WND, Cyc, etc.) support.

4.3.1.3. Document Representation. Bow approach is implemented without any generated features besides the words of the documents.

4.3.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

4.3.1.5. Machine Learning Algorithm. SVM with linear kernel is the machine learning module we use as the classification algorithm.

4.3.1.6. Feature Selection. No feature selection algorithm is used.

4.3.1.7. Performance Measures. Experiments are evaluated with microF and macroF scores.

4.3.2. Experiment Design and Results

The main motivation of the experiment is to implement the standard bow approach. the experiment yielded the microF score as 85.5 and macroF as 43.7 for the Reuters dataset.

4.3.3. Comments and Conclusion

We take the result achieved in the AW test as the benchmark score and try to improve it with the proposed algorithms in the following experiments.

4.4. Experiment1 - Introduction to Dependency Support in TC

In this experiment, we want to test the discriminative power of some of the widely used dependency types (object, subject, preposition, participles) as an extension to the bow approach with all words. Two parameters are varied in the experiments: the first parameter is about the form of the dependencies (stemmed or raw) in the feature vector, the second parameter determines whether dependencies are tested singly (e.g. object-verb as single feature type) or with an inter-combination (e.g. combination of object-verb, subject verb).

4.4.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

4.4.1.1. Datasets. Reuters is used as the main dataset for the experiments.

4.4.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Porter is used for stemming implementations.
- Feature Extraction: Dependencies are extracted by Stanford Parser. Experiments are performed without external knowledge base (WordNet, WND, Cyc, etc.) support.

4.4.1.3. Document Representation. Bow approach is extended with basic dependencies (object, subject, preposition, participles) extracted by Stanford Parser.

4.4.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

4.4.1.5. Machine Learning Algorithm. SVM with linear kernel is the machine learning module we use as the classification algorithm.

4.4.1.6. Feature Selection. No feature selection algorithm is used.

4.4.1.7. Performance Measures. Experiments are evaluated with microF and macroF scores.

4.4.2. Experiment Design and Results

Motivation of this experiment is to integrate syntactic information of the sentences with the traditional bow approach.

In this experiment, the bow approach is extended with the widely used dependency types (object-verb (obj), subject-verb (subj), prepositional modifier (prep), participle (part)) as seen in Figure 4.4. A sample sentence “The teachers organized the dinner” is processed according to alternative approaches in the figure. The dependen-

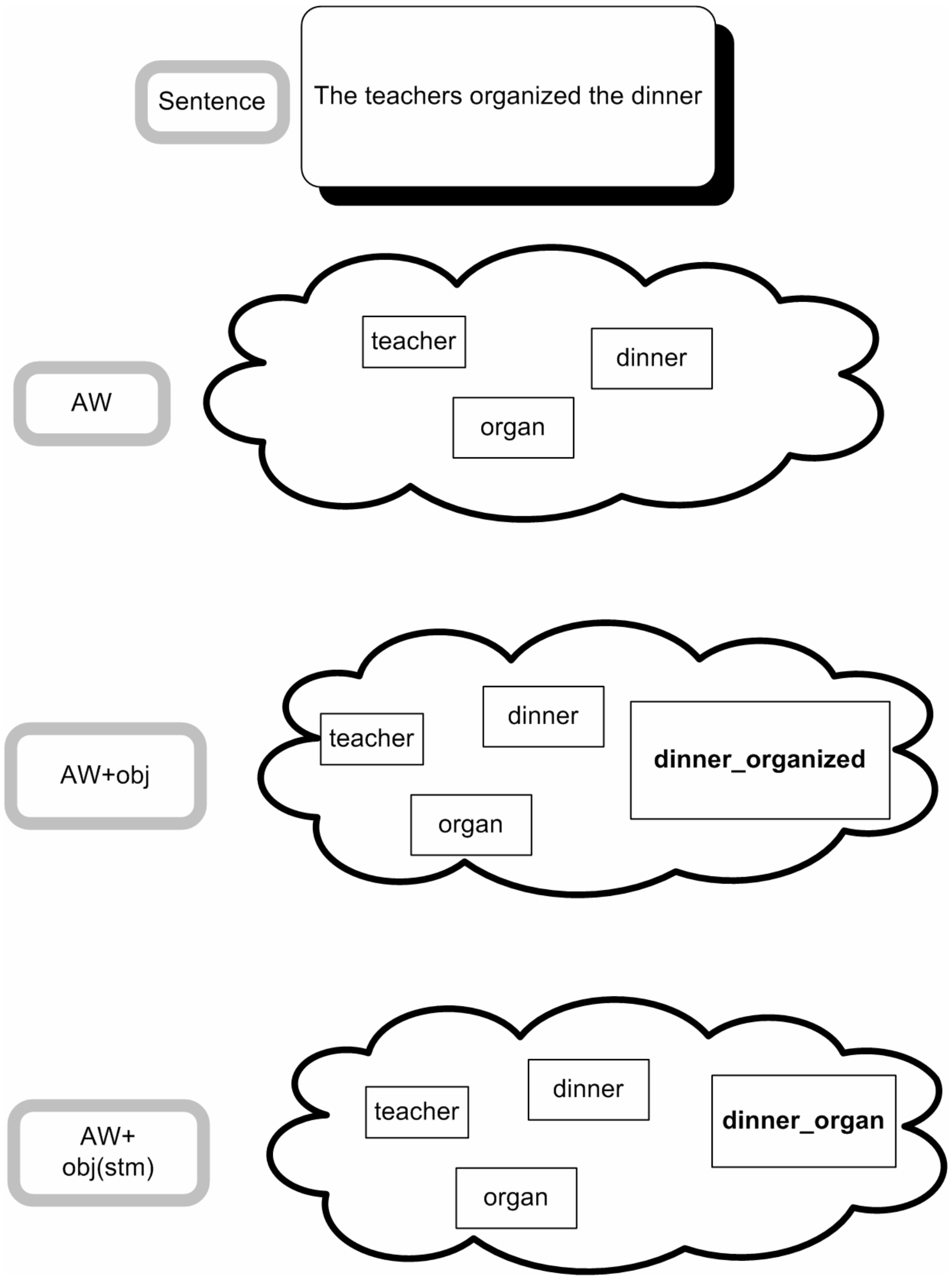


Figure 4.4. Feature types of Experiment 1

cies, success rates and number of features are summarized in Table 4.1. *AW* means standard use of SVM with all words (bow approach). *subj* stands for subject-verb dependency, *obj* is object-verb dependency, *part* stands for participle modifier and *prep* is prepositional modifier. *stm* denotes that stemmed form of the dependency is used. We list the results in Table 4.1.

Definition	**microF**	**macroF**	**FeatureNo**
AW	85.5	43.7	20674
AW+subj+obj	85.6	42.0	35940
AW+subj+obj (stm)	85.7	42.0	35247
AW+obj	**85.8**	**43.6**	**27096**
AW+obj (stm)	85.8	43.4	26583
AW+prep	85.6	42.9	35887
AW+prep (stm)	85.6	42.8	35326
AW+part	85.7	43.5	22088

Table 4.1. Success rates for different dependencies

4.4.3. Comments and Conclusion

As can be seen from Table 4.1, no significant change was observed in the success rates. This result is probably due to the limited type of experimented dependencies (only four dependencies were experimented with). Also too many dependency type features were observed in the experiment. For a dependency feature to reoccur in the dataset, both of the words in the word pair must be repeated with the same pattern which caused the repetition of this feature type harder than the word features (e.g. *report-price* dependency occurs much less than the independent occurrences of *report* and *price*). Using the stemmed forms of the dependencies (e.g. *report-price* instead of *reported-price*) did not decrease the feature vector size. This type of feature number increase caused more complexity with solution vector, which might also impact the success rates negatively.

4.5. Experiment2 - WordNet and WND Usage with Dependency Support

As an extension of the previous experiment set-up with some well known dependencies, our motivation in this part is to employ external knowledge resources like WordNet and its related domain oriented tool WND in addition to dependency use.

In this section, the main motivation is to explore the optimal use of related syntactic and semantic tools (WordNet, WND, Stanford Parser, etc.) and if possible, provide a generic, efficient and successful framework to enrich the bow approach in TC.

4.5.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

4.5.1.1. Datasets. Reuters is used as the main dataset for the experiments.

4.5.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Porter is used for stemming implementations.
- Feature Extraction: Dependencies are extracted by Stanford Parser. WordNet and WND are used as the external knowledge resources for the extraction of conceptual information.

4.5.1.3. Document Representation. Bow approach is extended with basic dependencies (object-verb) and POS (noun, verb) extracted by Stanford Parser and related concepts and domains extracted from WordNet and WND.

4.5.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

4.5.1.5. Machine Learning Algorithm. SVM with linear kernel is the machine learning module we use as the classification algorithm.

4.5.1.6. Feature Selection. No feature selection algorithm is used.

4.5.1.7. Performance Measures. Experiments are evaluated with microF and macroF scores.

4.5.2. Experiment Design and Results

By using the Stanford Parser, WordNet and WND, we extract new feature types (POS: noun, verb; Dependency: object-verb; semantic: synset id, synset id of the hypernmy of the synset) as mentioned in Section 4.2.

These new feature types are employed in the feature vector with two main alternatives:

- Single Approach : Each algorithm is employed by its own in alternative runs, so the feature vector consists of only those terms belonging to that specific algorithm
- AW Plus (AW+) Approaches : We append the extracted features of corresponding algorithms in addition to the AW features for the SVM algorithm, so the feature vector consists of all the words plus the new extracted terms specific to the algorithm.

Features extracted by WordNet and WND support are employed with the following alternatives:

- Domain Based (**Dmns**) : Wordnet and WND are used together to extract the related domains as keywords for our tests.

- Synset Based (**Syn**) : Synset based extraction employs only WordNet as lexical reference and extract synset id as keyword without utilizing WND.
- Hypernmy Based (**Hyp**) : Hypernmy property of WordNet is also employed by using the hypernmy of the corresponding synset to extract more generalized synsets.

The support of the Stanford Parser enables additional feature types to be used in the experiments in the following alternative forms:

- POS Based: POS information extracted from the Stanford Parser is employed for each word in each sentence in this approach. The widely-used POS types noun (**Nn**) and verb (**V**) are used in the experiment set-up.
- Dependency Based: Dependencies within words in each sentence are extracted in this approach. The most generic and experimented object-verb dependency is used. This method is also classified according to the utilization style.

  Couple (cpl) Usage: Dependency components are used together as one composite feature. e.g. *go-school* is used as one feature instance, which belongs to object-verb dependency type.

  Single Usage: Dependency components are used independently. e.g. *go* and *school* are used as two feature instances, which are oriented from object-verb dependency type.

Figure 4.5 summarizes the sample feature types of the experiment. Below, we additionally simulate the preprocessing phases of the sample sentence “The teachers organized the dinner” with AW+obj(stm,cpl) and AW+obj(dmn) options.

(i) Removal of non-alphabetic characters and case folding: “the, teachers, organized, the, dinner”

(ii) Elimination of stopwords: “teachers, organized, dinner”

(iii) Stemming: “teacher, organ, dinner”

(iv) Parsing (obj dependency extracted with stemming) : “teacher, organ, dinner,

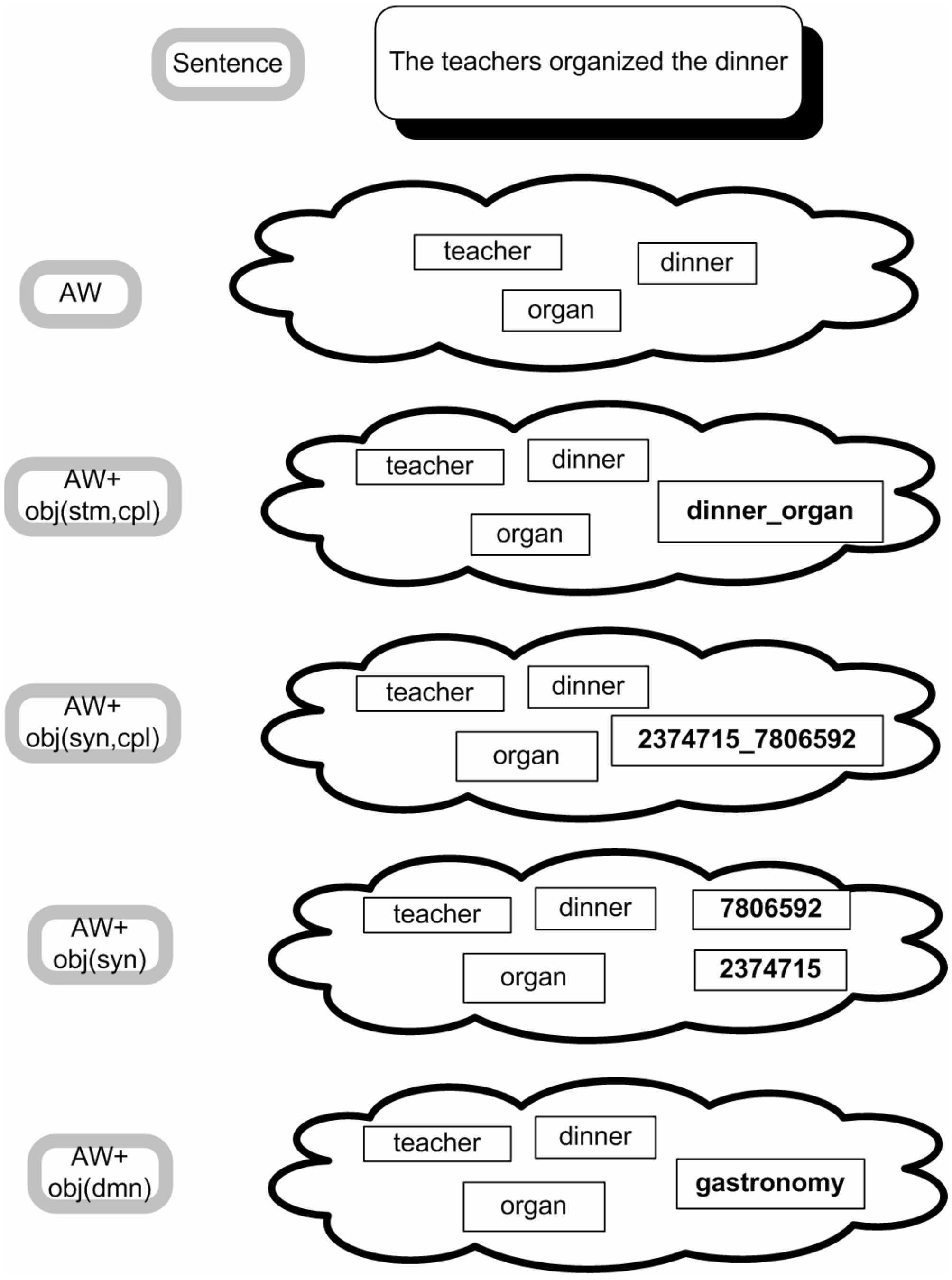


Figure 4.5. Feature types of Experiment 2 for a sample sentence

dinner_organ"

(v) Generating all features according to the option: "teacher, organ, dinner, dinner_organ" for *AW+obj(stm,cpl)* option and "teacher, organ, dinner, gastronomy" for *AW+obj(dmn)* option where *dinner* belongs to *gastronomy* domain but *organ* does not belong to a specific domain.

Table 4.2 is the result table for the tests containing the mentioned feature types with different combinations.

Methods	**microF/macroF**	**Keyword Number**
SINGLE APPROACHES		
AW	85.7/43.7	20674
Nn Dmn	46.6/5.5	136
Nn/V Dmn	46.4/5.4	140
Obj Cpl Dmn	49.1/10.0	10000
Obj Dmn	40.8/1.8	117
Nn/V Hyp Syn	73.4/32.4	117
Nn/V Syn	76.0/40.9	117
AW+ EXPERIMENTS		
Nn Dmn	81.0/36.8	20789
Obj Dmn	85.7/43.4	20753
Nn/V Syn	85.4/43.2	27786
Obj Cpl Syn	**85.8**/43.2	35183
Obj Syn	85.4/42.9	23856

Table 4.2. Success rates for the experiment: WordNet and WND usage with dependency support

4.5.3. Comments and Conclusion

Similar to the previous experiment, no significant improvement was observed in this experiment. WordNet lists the most common synset as the first one among all synsets for each word and we preferred to use these most common synsets, however, we still experienced ambiguity problem due to the possibility of other possible meanings of the words. On the other hand, WordNet synsets were analyzed to be more powerful than WND domains in achieving successful classification results. WND domains seemed to be too generic and not discriminative for the Reuters dataset.

As shown in Table 4.2, dependency based methods have shown better performance than POS based methods, which is a promising and original result among the related text classification approaches.

4.6. Experiment3 - Different Stemming Alternatives and WordNet Usage with Dependency Support

The main outcome of the previous section is that WordNet and WND use did not yield successful results in the TC problem. Also, if we compare the use of dependencies and POS (main outcomes of the Stanford Parser) in TC, lexical dependencies have shown more promising results.

In this section, we do not continue the implementation of WND and POS information, but rather focus on the dependency concept (including also a WordNet oriented alternative). In addition, we extend the syntactic analysis by including all possible dependencies. One of the main motivations of the current section is to compare the supplementary benefit of all possible dependencies. The other related aim is to analyze the optimal stemming algorithm for both raw words and dependency couples existing in the documents. In this part of the thesis, we question the effectiveness of the straightforward style of morphological stemming by analyzing lemmatizers in addition to the morphological stemmers.

4.6.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

Table 4.3. Dependencies and their examples

Symbol	dependency Type	Example Couples	Symbol	dependency Type	Example Couples
subj	subject-verb	they-break	**obj**	object-verb	glass-break
aux	auxiliary auxpassive	expected-are	**conj**	conjunctive	energy-petrochemical
attr	attributive	remain-year	**comp**	complement	decline-disclose
complm	complementizer	is-that, have-that	**mark**	mark	account-while
rel	relative	sell-of	**acomp**	adjectival complement	turn-bad
agent	agent	approve-bank	**adv**	adverbal clause modifier	quickly-open
rel	relative clause modifier	begin-season	**amod**	adjectival modifier	scientific-experience
infmod	infinitival modifier	way-invest	**rcmod**	relative clause modifier	begins-season
app	appositional modifier	monitoring-detection	**nn**	noun compound modifier	source-laser
poss	possession modifier	Asia-nations	**prt**	phrasal verb participle	cover-up
part	participle modifier	costs-related	**prep**	prepositional modifier	focus-research

4.6.1.1. Datasets. We use two datasets from the UCI machine learning repository: Reuters and NSF.

4.6.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. For both word and dependency features, stemming (lemmatizing) is performed with different alternatives: no stem, stemmed with Porter, stemmed with Stanford Parser and stemmed with combination of them.
- Feature Extraction: Dependencies are extracted by Stanford Parser. WordNet is alternatively used for synset identification.

4.6.1.3. Document Representation. 22 Dependencies are included in the feature vector independently and as an extension to the standard bow approach. Table 4.3 lists the related dependencies. Dependencies with number contents are eliminated (e.g. *num - numeric modifier*). The content of these features are so generic that their contribution will not be meaningful. Some of the similar dependencies are combined in the hierarchy (e.g. *dobj, iobj* and *pobj* as *obj*) in order to sum up their frequencies and discriminative power.

4.6.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

4.6.1.5. Machine Learning Algorithm. SVM with linear kernel is the machine learning module we use as the classification algorithm.

4.6.1.6. Feature Selection. No feature selection algorithm is used.

4.6.1.7. Performance Measures. Experiments are evaluated with microF and macroF scores.

4.6.2. Experiment Design

The main goal of this section is to compare the supplementary benefit of all possible dependencies and also analyze the optimal stemming algorithm for both raw words and dependency couples existing in the documents. For this purpose, we devise a two-stage analysis for our problem. We name the first and second stages as AW and AW+ Analysis, respectively.

Four stemming alternatives and one semantic oriented method are employed with the support of Porter Stemmer, Stanford Lemmatizer and WordNet utilization for both stages to be used in both AW Analysis and AW+ Analysis:

- Raw Form: No stemming process is implemented for the classification algorithm and the words are used in their raw forms.
- Only Porter Form: Morphological Porter stemmer is used for stemming.
- Only Stanford Form: Stanford lemmatizer is used for stemming.
- WordNet Synsets Form: After stemming by the Stanford lemmatizer, WordNet is employed to extract the synset variations of all the words. Porter stemmer is not implemented as an alternative because Porter output is not compatible with WordNet for two basic reasons. First, we need the correct POS information and

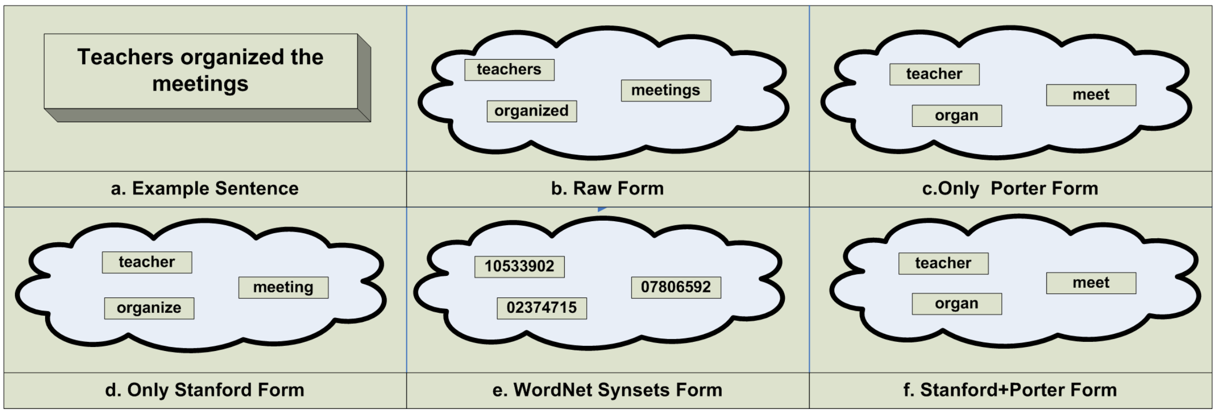


Figure 4.6. Sample keyword formations due to stemming alternatives for AW approach

the root form for the semantic analysis but this stemmer does not conserve the POS information of the derived word by extracting the possible shortest base form; and second, the outcome of this stemmer is not always in the standard base forms (e.g. *earli* instead of *early*, *continu* instead of *continue*, etc.).

- Stanford+Porter Form: Stanford lemmatizer is implemented initially for inflection removal, then Porter stemmer is used for the removal of derivational affixes of the same word.

4.6.2.1. AW Analysis. In this stage, the classical bow approach is implemented with the mentioned variations in order to find the optimal strategy for the bow approach. Representation of an example sentence in the solution vector is shown in Fig. 4.6. Fig. 4.6.b shows *Raw Form* for the example sentence without any stemming process. *Only Porter Form* and *Only Stanford Form* are shown in Fig. 4.6.c and Fig. 4.6.d, respectively. Fig. 4.6.e shows *WordNet Synsets Form* while *Stanford+Porter Form* is shown in Fig. 4.6.f.

4.6.2.2. AW+ Analysis. In the second stage, we perform the re-examination of the stemming alternatives; this time, not for the words but for the dependency couples. From another related perspective, we extend the bow approach by including the dependency couples which are varied by the stemming alternatives as shown in Fig. 4.7. For the required format of the bow approach for all words, we use the Porter form

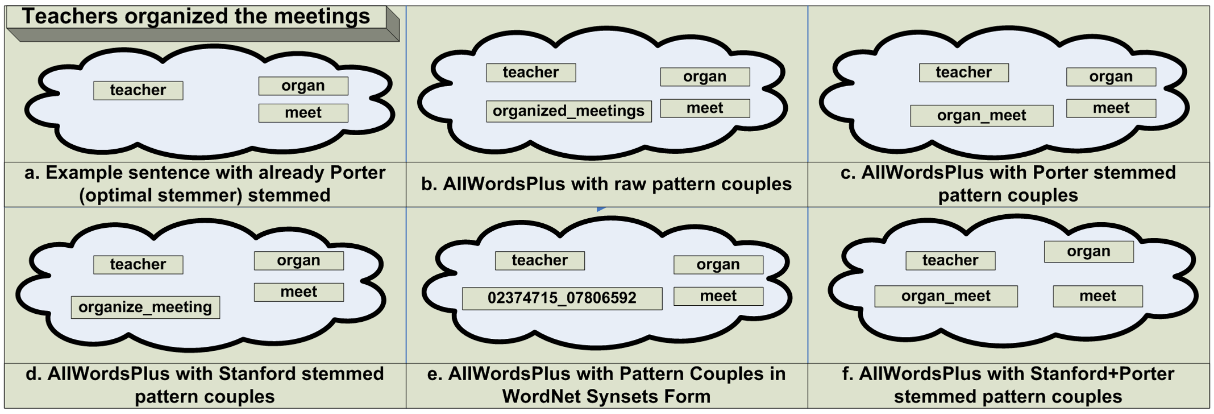

Teachers organized the meetings
teacher
organ
meet
a. Example sentence with already Porter (optimal stemmer) stemmed
teacher
organ
organized_meetings
meet
b. AllWordsPlus with raw pattern couples
teacher
organ
organ_meet
meet
c. AllWordsPlus with Porter stemmed pattern couples
teacher
organ
organize_meeting
meet
d. AllWordsPlus with Stanford stemmed pattern couples
teacher
organ
02374715_07806592
meet
e. AllWordsPlus with Pattern Couples in WordNet Synsets Form
teacher
organ
organ_meet
meet
f. AllWordsPlus with Stanford+Porter stemmed pattern couples

Figure 4.7. Sample keyword formations due to stemming alternatives for dependencies in AW+ approach

Table 4.4. All words stemming

	Reuters			NSF		
Approach	**Key#**	**microF**	**macroF**	**Key#**	**microF**	**macroF**
Raw	27094	85.63	43.86	21632	61.41	46.75
Only Porter	20292	85.58	43.83	14878	61.74	47.34
Only Stanford	23094	80.88	45.57	18062	61.21	46.02
WordNet Synsets	25202	80.62	44.87	21510	58.73	44.44
Stanford+Porter	18253	80.75	45.29	14186	61.74	47.42

as seen in Fig. 4.7.a. Fig. 4.7.b shows *Raw Form* utilization for the dependencies in addition to the optimally stemmed words for the example sentence. *Only Porter Form* and *Only Stanford Form* are used for dependency stemming as shown in Fig. 4.7.c and Fig. 4.7.d, respectively. Fig. 4.7.e represents *WordNet Synsets Form* while *Stanford+Porter Form* is shown in Fig. 4.7. Briefly, we use the optimally preprocessed *allwords* in the documents as the base keyword features for our algorithm and extend it with the dependency variations in each alternative implementation.

4.6.3. Results

4.6.3.1. AW Analysis Results. The results for AW analysis are shown in Table 4.4. As can be seen in the table, morphological stemming of Porter Stemmer with low keyword numbers and high success rates is found to be the optimal approach for stemming of the words in both datasets.

Table 4.5. Comparison of dependency forms

	Reuters			NSF		
Approach	**Key#**	**microF**	**macroF**	**Key#**	**microF**	**macroF**
Raw	27387	85.60	43.90	19534	61.91	47.21
Only Porter	27152	85.62	43.87	20631	61.90	47.15
Only Stanford	25561	85.60	43.85	19856	61.91	47.19
WordNet Synsets	26184	85.60	43.83	19892	61.92	47.20
Stanford+Porter	26693	85.61	43.82	20487	61.88	47.20

The raw form outperforms Porter stemmer in Reuters but falls far behind it in NSF with also many more keywords. Stanford lemmatizer, which is a more complicated stemmer with parser support, has mainly lower success rates and higher keyword numbers with respect to the Porter stemmer. The main reason for this difference is that the Stanford lemmatizer extracts many different forms for the same base form of the word because it only removes inflection, conserving the derivational affixes. For example, for the words *arrivals* and *arrived*, Stanford lemmatizer finds the base forms as *arrival* and *arrive*, respectively. On the other hand, Porter stemmer finds the same base form *arriv* for both words by removing both the derivational affixes and inflections.

4.6.3.2. AW+ Analysis Results. For AW+ approach, we can analyze the results from two points of view. The first view is the re-analysis of all the stemming approaches, this time within the dependency usage as shown in Table 4.5. We calculate the average of all results for each approach in both datasets. One of the outcomes of the analysis is that the differences between the stemming approaches is low, in terms of both number of keywords and success rate when compared with AW stemming alternatives (summarized in Table 4.4). A possible reason for this similarity is that dependency couples are integrated in only certain forms, which decreases the role of stemming. In parallel to this idea, the *raw* stemming process, which is the simplest form without any stemming process, yields slightly better scores than the others in both datasets.

For the other part of the analysis, we compare the dependency usage performance by analyzing the results through all stemming modes. Dependencies are ranked (*Rn*) according to their microF and macroF average scores (*avg*) with standard devia-

Table 4.6. Dependency performance ranks in descending order for Reuters and NSF

	Reuters	**Key#**	**microF**	**macroF**		**NSF**	**Key#**	**microF**	**macroF**
Rn	dependencies	avg	avg+/-std	avg±std	Rn	dependencies	avg	avg±std	avg±std
1	Part	25937	85,71±0,01	44,08±0,05	1	Adv	21112	62,22±0,11	47,51±0,01
2	Subj	29721	85,68±0,07	44,10±0,30	2	Comp	19975	62,24±0,12	47,49±0,04
3	Adv	28425	85,71±0,03	44,04±0,10	3	Cls	16227	62,00±0,04	47,53±0,04
4	Conj	32026	85,64±0,12	44,03±0,07	4	Part	18255	62,07±0,06	47,41±0,01
5	Poss	27401	85,68±0,03	43,89±0,04	5	Poss	16186	61,95±0,08	47,52±0,01
6	Amod	31690	85,66±0,06	43,91±0,12	6	Mark	15672	61,89±0,02	47,48±0,03
7	Rcmod	26673	85,60±0,02	43,94±0,04	7	Conj	29144	62,00±0,09	47,35±0,20
8	Agent	21603	85,63±0,01	43,91±0,04	8	Complm	15386	61,83±0,03	47,46±0,01
9	App	24676	85,61±0,02	43,91±0,02	9	App	15993	61,83±0,03	47,41±0,00
10	Comp	34414	85,73±0,02	43,78±0,13	10	Prt	15018	61,88±0,05	47,35±0,01
11	Obj	34907	85,67±0,09	43,79±0,08	11	Rcmod	18365	61,81±0,03	47,36±0,00
12	Acomp	20705	85,59±0,01	43,86±0,00	12	Infmod	15417	61,81±0,00	47,34±0,00
13	Attr	20378	85,58±0,00	43,83±0,00	13	Rel	16250	61,77±0,10	47,35±0,02
14	Cls	24202	85,55±0,01	43,86±0,01	14	Agent	15529	61,77±0,04	47,33±0,01
–	**Benchmark**	**20292**	**85,58±0,00**	**43,83±0,00**	15	Attr	14892	61,74±0,00	47,34±0,00
15	Complm	21442	85,57±0,01	43,83±0,02	–	**Benchmark**	**14878**	**61,74±0,00**	**47,34±0,00**
16	Prt	21122	85,55±0,03	43,84±0,03	16	Acomp	15035	61,67±0,00	47,36±0,04
17	Infmod	21922	85,54±0,01	43,82±0,00	17	Amod	27535	62,13±0,16	46,83±0,06
18	Mark	22638	85,53±0,01	43,82±0,01	18	Obj	27737	61,88±0,13	47,06±0,33
19	Rel	20977	85,52±0,05	43,80±0,05	19	Subj	29355	62,28±0,17	46,49±0,02
20	Prep	33487	85,71±0,05	43,55±0,16	20	Prep	31546	62,15±0,22	46,35±0,33
21	Nn	31220	85,45±0,07	43,66±0,04	21	Nn	27726	61,79±0,09	46,23±0,06
22	Aux	29528	85,45±0,04	43,50±0,27	22	Aux	19390	61,19±0,17	46,60±0,44

tions (*std*) according to the alternative stemming modes in Table 4.6. As can be seen from the table, 14 dependencies in Reuters and 15 dependencies in NSF out of the possible 22 dependency types, have the power of outperforming the benchmark. The critical point is that nine qualified dependencies that increase the benchmark, achieve this performance in both datasets consistently with also very low standard deviations. Out of these nine *positive* dependencies, we focus on the highest four dependencies, namely : *part-participle modifier*, *adv-adverbal clause modifier*, *conj-conjunctive* and *poss-possession modifier.*

Fig. 4.8 shows the incremental effect of these dependencies in Reuters and NSF. This improvement is shown by the dark colored part over the light gray color of the benchmark score for each dependency in the figure. These dependency types, when utilized standalone, improve the benchmark by around 0.4%-0.5%.

There is also consistency for the three most unsuccessful dependency types in both Reuters and NSF: *preposition modifier - prep*, *noun compound modifier - nn*, and

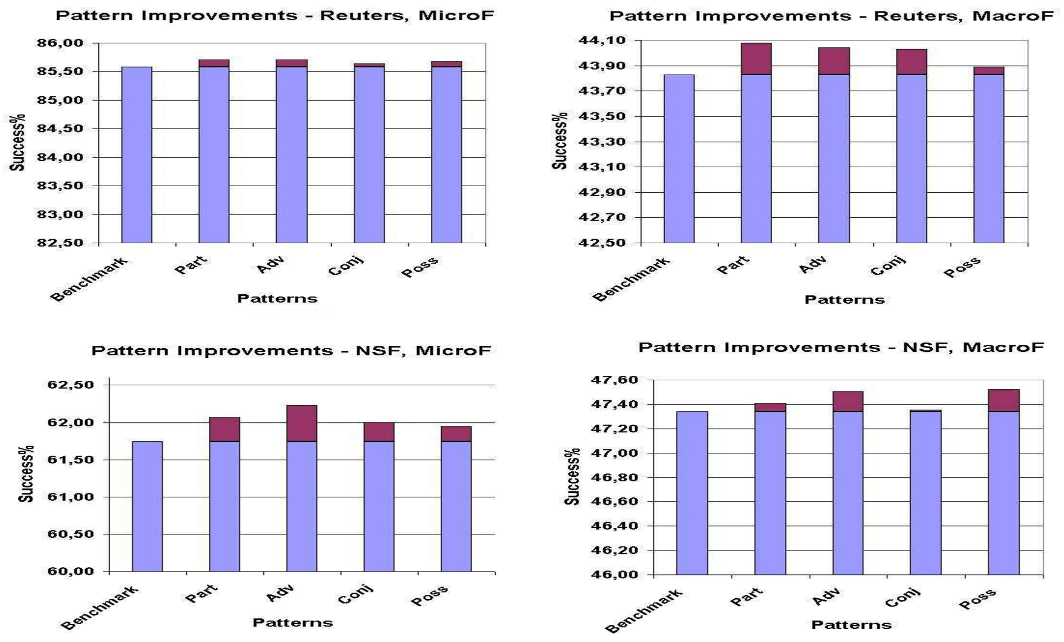


Figure 4.8. Improvements of the successful dependencies over the benchmark

auxiliaries-aux. The *aux* dependency gives the worst results for both datasets in both microF and macroF values. The main factor for this failure seems to be the fact that the relationship within the dependency structure is generic (e.g. : make-is, running-are) which is not feasible for the classification problem. The *nn* dependency is the second worst dependency with low scores in both microF and macroF measures. The *prep* dependency is an interesting dependency with leading scores in microF but lowest results with macroF in both datasets. There are many sub-dependencies of it (e.g. prep-of, prep-with) that may confuse the context and affect the success rates.

4.6.4. Comments and Conclusion

Main outcome of this section is the result that the benchmark of the classical approach in TC is improved by the support of dependency utilization. The most successful dependency types improve the benchmark by around 0.4%-0.5%.

Another contribution of this study is its approach in stemmer utilization. Stemming is analyzed not only for the words but also for all the extracted dependency couples in the texts. Porter stemming is observed to be the optimal stemmer for all words, while the raw form without stemming slightly outperforms the other approaches in dependency stemming.

One of the main observations of the experiments is that most of the dependency types (see Table 4.3) yielded many instances (distinct word pairs) for a dataset that also caused an excessive number of features with mostly zero or quite low frequencies in most of the documents. This factor causes the solution vectors of the machine learning algorithms to be highly sparse which points out a feature filtering / selection implementation to be an explicit need for the proposed system.

In the next chapter, we will deal with pruning analysis of features as the basic feature selection algorithm and perform further analysis of dependencies (e.g. combination of the leading ones) in text classification.

5. DEPENDENCY USAGE AND PRUNING IN TC

In the previous chapter, we analyzed that support of specific dependency types in addition to the all words improves the performance of the benchmark bow approach. The improvement was analyzed to be low with a significant increase in the number of features. This inefficient usage was stated to be due to the sparsity of the solution vector (caused by the large number of dependencies with low frequencies) and the lack of further analysis of dependencies.

In this chapter, we present a comprehensive analysis of the lexical dependency and pruning concepts for the text classification problem. Dependencies are included in the feature vector as an extension to the standard bow approach. The pruning process filters features with low frequencies so that fewer but more informative features remain in the solution vector. We analyze in detail the pruning levels for words, dependencies, and dependency combinations for different datasets. The main motivation in this chapter is to make use of dependencies and pruning efficiently in text classification and to achieve more successful results using much smaller feature vector sizes. We used three independent datasets in the experiments and obtained statistically significant improvements for most of the proposed approaches.

5.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

5.1.1. Datasets

We use three datasets from the UCI machine learning repository: Reuters, NSF and MiniNg20 [4]. We choose datasets with different characteristics in order to be able to analyze the effect of the proposed methods on different types of documents and to make comparisons between them.

5.1.2. Preprocessing Operations

Preprocessing is performed with standard routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Stemming of the words is implemented by Porter Stemmer.
- Feature Extraction: Dependencies are extracted by Stanford Parser without stemming. Experiments are performed without external knowledge base (WordNet, WND, Cyc, etc.) support.

5.1.3. Document Representation

36 Dependencies are included in the feature vector independently and as an extension to the standard bow approach. Combination of the leading dependencies are also experimented.

5.1.4. Term Weighting Approach

We choose the standard tf-idf metric for term weighting in our methods.

5.1.5. Machine Learning Algorithm

SVM with linear kernel is the machine learning module we use as the classification algorithm.

5.1.6. Feature Selection

Basic feature selection algorithm (pruning) is implemented based on the term's frequency in the whole dataset. Optimization of the pruning levels with different feature types (word, dependency) and different datasets are performed.

5.1.7. Performance Measures

Experiments are evaluated with microF and macroF scores. Micro sign test, macro sign test and micro sign test with positive instances are performed to measure the significance of the improvement.

5.2. System Modules

In this section, we discuss the details of the main utilities proposed in the experiment: pruning implementation and the dependency usage.

5.2.1. Pruning Implementation

We use pruning in order to filter low-frequency features so that fewer but more informative features remain in the final solution vector. This process is implemented by eliminating the terms that occur less than a certain threshold value in the whole training set. This threshold is named the *pruning level* (PL). We analyze the pruning levels for words, dependencies, and dependency combinations separately. PL=n ($n \geq 1$) indicates that features occurring less than n times in the training set are filtered, thus only the features with at least n occurrences are used in the solution vector. Note that PL=1 means that no pruning is implemented for that feature type.

The pruning concept is especially useful for dependency features. As mentioned in Section 1, a dependency is formed by combining any two dependent words. Most of the dependency types (see Table 5.1) yield many instances (distinct word pairs) for a dataset. This causes an excessive number of features. Moreover, for a dependency feature to reoccur in the dataset, both of the words in the word pair must be repeated with the same pattern. This indicates that the majority of these features have zero or quite low frequencies in most of the documents [39]. This makes the solution vectors used in the machine learning algorithms highly sparse. As will be seen later, pruning such features has a significant effect on both the accuracy and the efficiency of the methods.

One of the main contributions of this study is that we perform parameter tuning by analyzing different values for each method and dataset to reach the optimal PL values. These methods and the details of the pruning analysis will be explained in Section 5.3.

5.2.2. Dependency Analysis

36 dependency types are used in the tests. Table 5.1 shows the complete list of dependency types accompanied with their definitions and some examples. Dependency types formed of numeric tokens (e.g. *numeric modifier*) were eliminated because they did not improve the accuracy of the system in the experiments. Some of the similar dependency types were combined in order to sum up their frequencies and thus increase the discriminative power of the classifier. For instance, the types *dobj*, *iobj* , and *pobj* that denote dependencies formed of the indicated object and the main verb of a sentence yield many overlapping instances and thus they were considered as a single dependency type (*obj*).

5.3. Experiment Design

We designed an incremental framework for the analysis of the dependency and pruning concepts in the TC domain. As can be seen in Figure 5.1, the framework consists of four main stages. At each stage, the method of the preceding stage is improved by adding a new property in order to increase the overall performance of the system. The details of the stages are explained in the following subsections.

In the methods where pruning is applied, we repeat the experiments with incremental PL values. In order to discover the optimal value, we stop incrementing the PL value when the success rates start to drop consistently. Pruning for words and dependencies were analyzed separately since the optimal pruning levels will be different in each case. Since dependencies are formed as pairs of words, they occur with much less frequencies than words and thus they are expected to be optimized at smaller PL values.

Table 5.1. Dependency types used in the experiments

Symbol	Type	Examples
acomp	adjectival complement	turn-bad, make-clear
adv	adverbal clause modifier modifier	quickly-open, also-plan
agent	agent	approve-bank, approach-vector
amod	adjectival modifier	scientific-study, principal-investigator
app	appositional modifier	monitoring-detection, eigenvalues-separation
attr	attributive complement	remain-year, payable-april
aux	auxiliary passive	expected-are, study-to
cls	clause modifier	use-determine, determine-interact
comp	complement	decline-disclose, plan-study
complm	complementizer	is-that, make-that
conj	conjunctive	energy-chemical, variables-observations
infmod	infinitival modifier	way-invest, project-study
mark	mark	account-while, although-beginning
nn	noun compound modifier	source-laser, detection-problem
obj	object-verb	glass-break, study-questions
part	participle modifier	costs-related, measurements-needed
poss	possession modifier	Asia-nations, their-regulations
prep-along	prepositional modifier - along	moves-chromosomes, come-way
prep-as	prepositional modifier - as	farming-strategy, treat-human
prep-at	prepositional modifier - at	available-institution, glass-table
prep-btwn	prepositional modifier - between	relation-algebra, black-white
prep-by	prepositional modifier - by	displayed-species, performed-actor
prep-for	prepositional modifier - for	use-study, hunt-food
prep-from	prepositional modifier - from	show-studies, come-home
prep-in	prepositional modifier - in	low-cost, holiday-june
prep-into	prepositional modifier - into	extend-regions, divide-parts
prep-none	prepositional modifier - generic	clarify-by, prevent-from
prep-of	prepositional modifier - of	modeling-behavior, problems-students
prep-on	prepositional modifier - on	work-project, put-table
prep-over	prepositional modifier - over	stayed-time, talk-subjects
prep-to	prepositional modifier - to	similar-theory, seem-me
prep-with	prepositional modifier - with	vary-depth, gone-wind
prt	phrasal verb participle	cover-up, pointed-out
rcmod	relative clause modifier modifier	begins-season, type-large
rel	relative modifier	which-allows, numbers-large
subj	subject-verb	they-break, student-studies

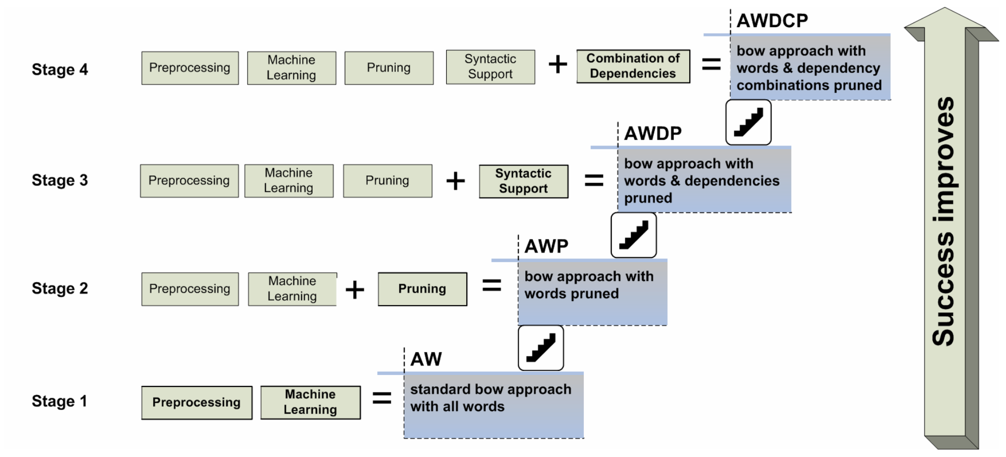


Figure 5.1. General system architecture with the proposed methods

We see that the effect of pruning diminishes at higher pruning levels since most of the features have already been pruned at earlier levels. For instance, while pruning the dependencies on the MiniNg20 dataset, increasing the PL value from PL=1 to PL=2 eliminates about 75% of the dependencies, indicating that most of the dependency pairs occur only once in the whole dataset. On the other hand, when we increment the PL value, for instance, from PL=20 to PL=30, only one dependency is pruned among the dependencies in the solution vector with PL=20. The same situation occurs during word pruning and on the other datasets too. Based on this observation, we performed the pruning analysis with small increments in initial pruning levels (e.g. PL=1, 2, 3) and larger increments in higher levels (e.g. PL=20, 30, 50).

5.3.1. AW

AW (all words) is the benchmark method that uses the standard bow approach with all the words in the feature vector. It is implemented once for each dataset without any variation. Our main motivation in this study is to extend this approach by the proposed solutions and outperform it in terms of success rate and feature vector size.

5.3.2. AWP

The AWP (all words with pruning) method considers all the words in the document collection, but filters them by the pruning process. Algorithms that are similar to AWP have already been experimented in TC, but they lack a detailed analysis of alternative pruning levels [38]. We implement this method with several pruning levels (2, 3, 5, 8, 13, 20, and 30) to determine the optimal word PL value for each dataset.

5.3.3. AWDP

The AWDP (all words and dependencies with pruning) method extends both the AW and the AWP approaches by using dependencies in addition to words and also by pruning both feature types to obtain the final feature set. The PL values for words are fixed at the optimal values found by the AWP method. The dependencies corresponding to a dependency type are generated and they are filtered using varying pruning levels. Then the classification algorithm is executed using the pruned feature vector. This process is repeated separately for each dependency type. The main motivation of this method is to perform pruning level analysis for dependencies. We use the PL values (2, 3, 5, 8, 13, 20, and 30) in this stage.

5.3.4. AWDCP

The AWDCP (all words and dependency combinations with pruning) method extends the AWDP method by using the combination of the leading dependencies instead of using them individually. For this purpose, the five most successful dependency types are selected and used together for each dataset. We perform pruning level analysis using 10 different pruning levels: 2, 3, 5, 8, 13, 20, 30, 50, 80, 120. Different from the previous methods, the PL value was increased up to PL=120 since the success rates continued to improve past PL=30 for some of the experiments. To the best of our knowledge, this is the first study that considers the successful dependency types and uses them as an extension to the bow approach in the TC problem.

5.4. Analysis of Results

In this section, we first explain the optimal values of the pruning level parameter used in the methods. Then we explain the results of the experiments with these parameter values. Following this, we focus on some specific aspects of the methods and the experiments and comment on these: pruning level analysis, optimal feature number, significance of the improvements, and dataset comparison.

5.4.1. Optimal Parameter Decisions

The AW method uses the standard bow approach and does not involve any pruning. For the AWP method, the optimal word pruning level was found as 13 among the experimented values for all the three datasets. As stated previously, words and dependencies are pruned independent of each other in the AWDP method. The PL value for

Table 5.2. Success scores of the proposed methods

	Reuters			NSF			MiniNg20		
	Key#	**microF**	**macroF**	**Key#**	**microF**	**macroF**	**Key#**	**microF**	**macroF**
AWDCP	4138	86.03	45.26	3908	66.01	47.68	2914	54.23	51.65
AWDP	4198	85.96	45.07	2829	65.07	47.10	3114	54.13	51.53
AWP	3976	85.84	44.85	2478	64.58	46.49	2863	53.62	51.02
AW	20292	85.58	43.83	13424	64.46	46.11	30970	46.42	43.44

Table 5.3. Success scores of the leading dependencies in AWDP with the optimal PL values

	Reuters	**PL**	**word:13**	**NSF**	**PL**	**word:13**	**MiniNg20**	**PL**	**word:13**
		PL	**dep.:8**		**PL**	**dep.:8**		**PL**	**dep.:2**
		microF	**macroF**		**microF**	**macroF**		**microF**	**macroF**
1	prep-in	85.96	45.07	nn	65.07	47.10	prt	54.13	51.53
2	prep-from	85.87	45.14	**amod**	65.03	47.09	rel	54.04	51.45
3	**amod**	85.93	45.04	subj	64.97	46.83	app	53.97	51.33
4	part	85.93	45.04	obj	64.79	46.82	infmod	53.97	51.33
5	**comp**	85.99	44.94	**comp**	64.78	46.76	prep-btwn	53.87	51.34

words was fixed as 13 (as determined in the previous stage) and among the PL values analyzed, optimal dependency pruning levels were found as 8, 8, and 2 for Reuters, NSF, and MiniNg20, respectively. These dependency PL values are the optimal values corresponding to the dependency type that gave the best success rate in each dataset (e.g. *prep-in* in Reuters). However, we observed in the experiments that most of the successful dependency types converge to similar optimal pruning levels.

For the AWDCP method, the five leading dependency types determined by AWDP for each dataset (see Table 5.3) were considered. All the dependencies formed of these five dependency types were included in the feature vector and the method was tested with varying pruning levels. The optimal PL values for dependency combinations were determined as 50, 8, and 8 for Reuters, NSF, and MiniNg20, respectively. The success ratios of the methods as a function of the PL values will be compared and analyzed in Section 5.4.3.

5.4.2. Performance of the Methods

Table 5.2 shows the success rates of the methods in terms of their microF and macroF scores. The results shown represent the most successful result obtained for each method under the optimal PL value. The AW method that we use as the benchmark method for comparing with other methods yields the worst results. The AWP method outperforms the baseline performance when applied with the optimal word PL values. Similarly, the performance of AWP is exceeded by the AWDP method. The success rates shown in the table for AWDP correspond to the results obtained using the optimal dependency PL values and the most successful dependency type (see Table 5.3). In fact, as can be seen from the tables, using any one of the five best dependency types gives more successful results than using only words.

Table 5.3 shows the performance of AWDP for the best dependency types for each dataset. The AWDCP method, which is the most sophisticated approach proposed in this study, incorporates all the dependencies formed of these dependency types as features in the feature vector. As a result, it outperforms all the other methods with the optimal pruning levels for dependency combinations.

5.4.3. Pruning Level Analysis

Figures 5.2 and 5.3 show the microF and macroF scores as a function of PL (AWP with PL=1 corresponds to AW). The horizontal axes in the figures correspond to the word PL values for AWP, dependency PL values for AWDP (word PL value is fixed to the optimal value), and dependency combination PL values for AWDCP (word PL value is fixed to the optimal value). With both microF and macroF measures and in almost all PL values, AWDP improves the success rate of AWP and AWDCP gives the best results for all the datasets. This result is consistent with the analysis discussed in Section 5.4.2.

We see that the curves in the figures follow a similar pattern with respect to the PL improvement. Although the optimal pruning level varies depending on the

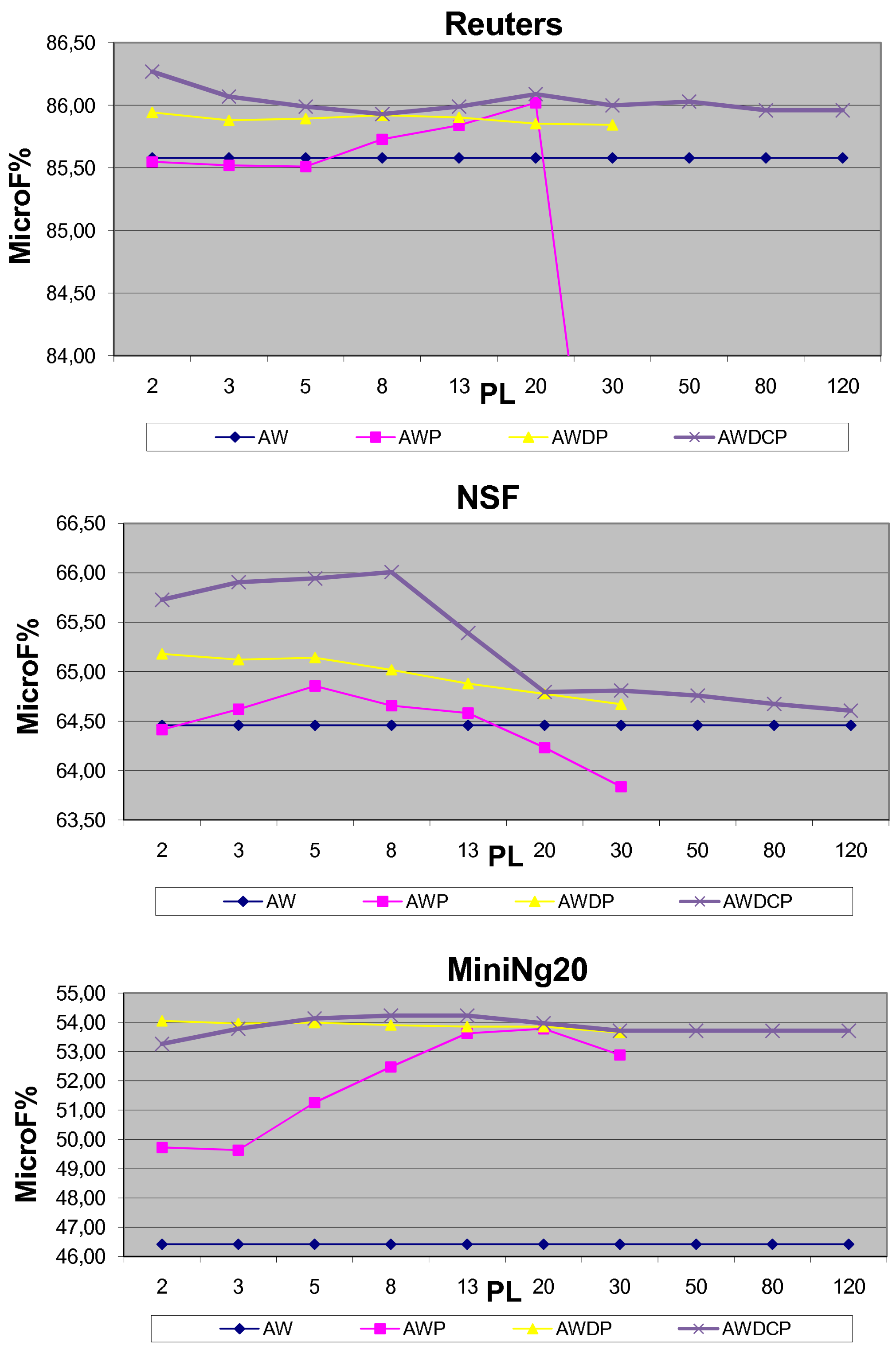


Figure 5.2. Pruning level analysis with the proposed methods (microF)

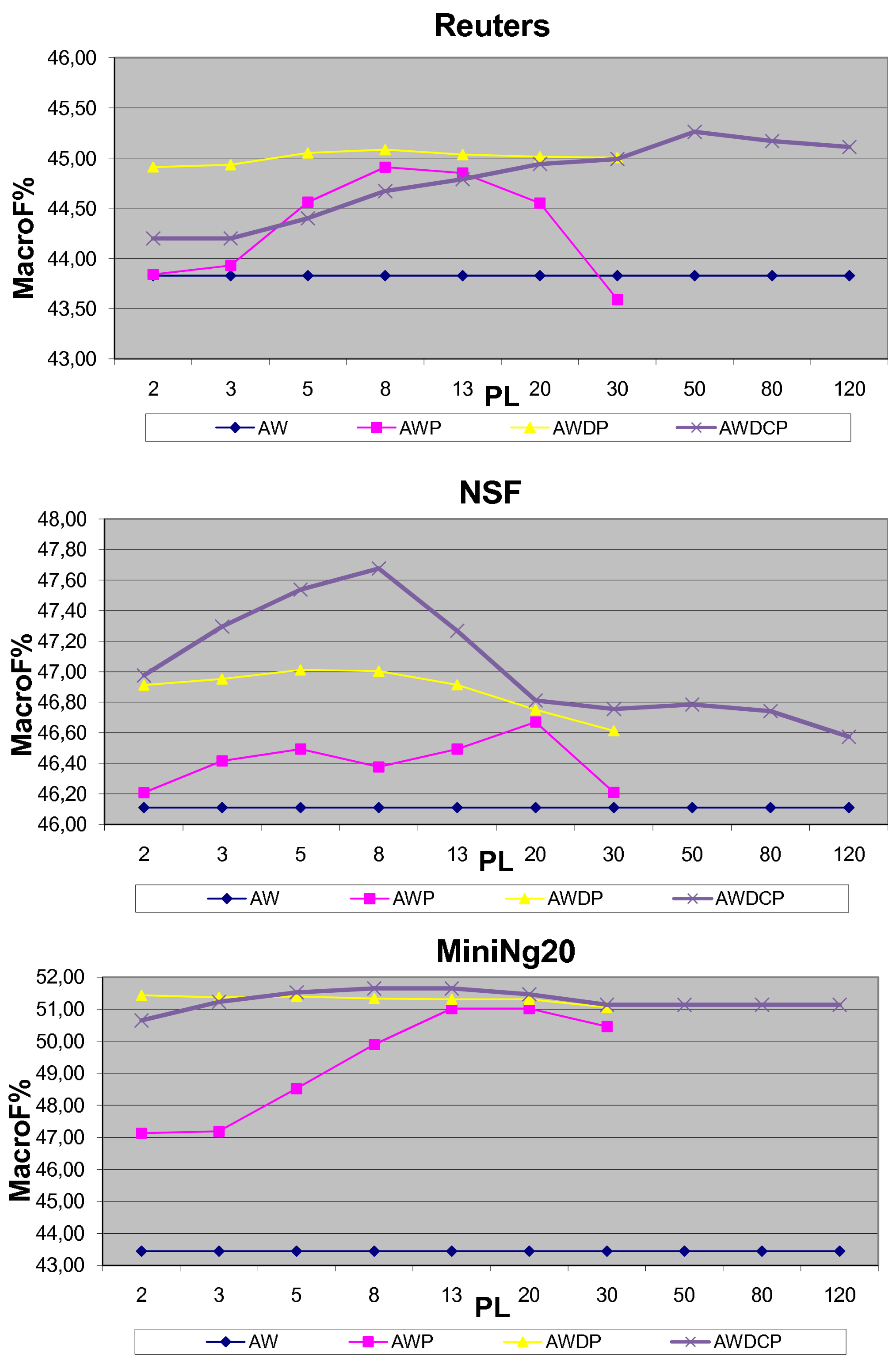


Figure 5.3. Pruning level analysis with the proposed methods (macroF)

method and the dataset, each performance curve is bell-shaped and the success scores first increase up to the optimal PL value and then decrease. This analysis reveals the fact that the pruning process arrives at fewer but more informative features for TC at some PL value and after this optimal level the process starts to eliminate rare but informative features which causes the performance to fall.

5.4.4. Optimal Feature Numbers

Table 5.2 shows the number of keywords for each method and dataset. Since it does not involve any pruning process, the AW method uses all the words in the dataset in the feature vector. For the other approaches, the keyword numbers are seen to be between 2400 and 4200.

In different studies related to feature selection in the literature, several feature number levels (500, 1000, 2000, 5000, 10000, etc.) were reported to give successful results with different machine learning algorithms [6, 56, 5]. The optimal feature number range that we obtained in this work (2400-4200) can be said to be consistent with these stated results.

5.4.5. Significance of the Improvements

Table 5.4 compares the methods used in this work and shows the statistical improvement results. Three significance tests that have been defined in Section 2.7.2 were applied. The micro sign test measures the improvement over the whole document-category matrix. We have derived an extended version of this measure (micro sign test with positive instances) in order to avoid sparsity, which focuses on only the positive samples in this matrix. The macro sign test is category oriented and it considers the F-scores of the two systems for each category of the datasets. The methods were compared according to the z values and the corresponding confidence areas were checked in the z-table. We use the symbols and terminology defined in Section 4.1.7 to denote the result of a comparison:

Table 5.4. Statistical comparison of the proposed methods - AW, AWP, AWDP and AWDCP

	Micro Sign, All	**Micro Sign, +**	**Macro Sign**
Reuters			
AWP over AW	$\sim$	$>$	$\gg$
AWDP over AWP	$\sim$	$\sim$	$\sim$
AWDP over AW	$>$	$\gg$	$>$
AWDCP over AW	$>$	$\gg$	$\gg$
AWDCP over AWP	$>$	$\sim$	$\sim$
AWDCP over AWDP	$\sim$	$\sim$	$\sim$
NSF			
AWP over AW	$\sim$	$>$	$\sim$
AWDP over AWP	$\gg$	$\gg$	$>$
AWDP over AW	$\sim$	$\gg$	$>$
AWDCP over AW	$\gg$	$\gg$	$\gg$
AWDCP over AWP	$\gg$	$\gg$	$\gg$
AWDCP over AWDP	$\gg$	$\gg$	$\gg$
MiniNg20			
AWP over AW	$\gg$	$\gg$	$\gg$
AWDP over AWP	$\sim$	$\sim$	$\sim$
AWDP over AW	$\gg$	$\gg$	$\gg$
AWDCP over AW	$\gg$	$\gg$	$\gg$
AWDCP over AWP	$\sim$	$\sim$	$\sim$
AWDCP over AWDP	$\sim$	$\sim$	$\sim$
All datasets			
AWP over AW	$\sim$	$\gg$	$\gg$
AWDP over AWP	$\gg$	$\gg$	$\gg$
AWDP over AW	$\gg$	$\gg$	$\gg$
AWDCP over AW	$\gg$	$\gg$	$\gg$
AWDCP over AWP	$\gg$	$\gg$	$\gg$
AWDCP over AWDP	$\gg$	$\gg$	$>$

From the table, it can be seen that in most cases there is a significant improvement between a method and its predecessor. The results of the AWDCP method are always statistically better than the benchmark (AW) method in all the datasets. The last part of the table shows the overall results by taking into account all the instances from the three datasets. We observe that each method significantly outperforms its predecessor method (AWDCP >> AWDP >> AWP >> AW) and AWDCP (the most advanced method proposed so far) is significantly the best method.

5.4.6. Dataset Comparison

We performed the experiments with three different datasets: Reuters, NSF and MiniNg20. Reuters and NSF can be regarded as alike with many mutual characteristics, while MiniNg20 differs from them in terms of formality, skewness, and other related issues. In this section, we analyze the results from the dataset perspective and compare the datasets according to these characteristics and results.

5.4.6.1. Skewness Factor. A point that is worth noting is the difference between the microF and macroF scores in a dataset. As can be seen in Table 5.2, the microF score of Reuters is about 1.9 times of its macroF score and this ratio is about 1.4 in NSF. On the other hand, the MiniNg20 dataset yields similar microF and macroF scores in almost all experiments. As explained in Section 2.7.2, the microF measure gives equal weight to each document. In the case of the macroF measure, equal weight is given to each category, which favors the documents in rare categories (categories including a small number of documents). Based on this fact, the category-document distribution (skewness factor) becomes an important factor for microF-macroF comparison.

In skewed datasets, there are not sufficient number of documents in some of the classes, which causes the macroF measure to drop significantly. Reuters is a highly skewed dataset; NSF is also skewed but its skewness is less than Reuters. On the other hand, MiniNg20 is a balanced dataset and similar microF and macroF values can be obtained.

5.4.6.2. Optimal PL Values. The optimal PL value shows variation as a function of the applied method and the dataset. When we compare the pruning levels with respect to the methods, we observe an expected pattern. The PL values of AWDP are less than those of AWP. Since dependencies are formed of pairs of words, their frequencies in the dataset are lower than the frequencies of words, which in turn requires a lower dependency PL value to eliminate the irrelevant dependencies. A similar behavior exists between the AWDP and AWDCP methods. The feature vector in AWDCP includes all the dependencies corresponding to five leading dependency types rather than dependencies of a single type as in the case of AWDP. Increasing the number of dependency types causes more dependencies (features) in the solution vector. Thus higher pruning levels are needed to eliminate the irrelevant dependencies and reach the optimal feature number.

The dataset type also has an effect on the pruning levels. The word PL values for the three datasets reach their maxima at similar points and we fixed the optimal values as PL=13. For dependencies, the optimal PL value of MiniNg20 is much less than the values of Reuters and NSF. This is due to the informal writing style and misspellings in this dataset, which makes it difficult to find lots of repeated occurrences of a word pair. Thus, a low PL value is sufficient to filter most of the irrelevant dependencies. When we apply the AWDCP method, the PL values of Reuters and MiniNg20 increase significantly (from 8 to 50 and from 2 to 8, respectively), while it stays the same (8) for NSF. This is understandable because in the AWDP tests with NSF, we may have selected the PL value alternatively as 3 or 5 (which, in the next stage, would yield the improvement of AWDCP) but we have preferred the larger value (PL=8, which gives a success rate similar to PL=3 and PL=5) to decrease complexity. This shows that the NSF results are compatible with the other datasets in terms of PL optimality. Higher PL values in the AWDCP method is closely related with the idea of the optimal feature number that has been mentioned in the above paragraph. There are additional possible feature types (so more features) with AWDCP so more pruning implementation is needed to reach the optimal feature number that gives the most successful results.

5.4.6.3. Formality Level. The Reuters and NSF datasets can be stated to have a formal style, whereas MiniNg20 is mostly informal. Since the efficiency of parsing is directly effected by the grammatical level of a document, we achieve less accurate parse results in MiniNg20 due to morphological and syntactic errors. There are many misspellings and related text errors in MiniNg20 which decreases the success rate of classification: about 60% of the words and 70% of the dependencies occur only once in the whole dataset and are eliminated when PL=2. As can be seen by a comparison of AW and AWP in Table 5.2, this initial pruning process increases the success rates in MiniNg20 much more than Reuters and NSF, which shows the success of pruning especially in informal datasets.

5.4.6.4. Common Successful Dependencies. Table 5.3 shows that Reuters and NSF have two common dependencies (shown in bold) in the five leading dependencies, while MiniNg20 has no common dependencies. One of the common dependencies is *comp* which is a structurally complicated dependency formed by integrating two verbs that have the same subject in the adjacent clauses. However, in the informal MiniNg20 dataset, this complex dependency does not improve the performance of the classifier due to the simple and ungrammatical sentence structures in the dataset. Instead of it, *prt* (phrasal verb participle, e.g. *write down*) is one of the simplest dependencies and yields the most successful results with MiniNg20.

5.5. Comments and Conclusion

The main motivation of this chapter was to extend the standard bow method used in TC by extracting fewer but informative features, so that more successful results can be achieved with much less features. For this purpose, we incorporated the concepts of lexical dependencies and pruning into the algorithms and determined the optimal parameter values for each.

36 dependencies and 10 PL values were experimented in four main methods (AW, AWP, AWDP, AWDCP). AW is named the standard bow approach and each of the

other three methods is an extended version of its predecessor, improved by dependency and pruning support under the optimal parameter settings. We used SVM for the machine learning component, which is a state-of-the-art classifier in TC, and the Stanford Parser as the syntactic tool. We repeated all the experiments in three different datasets (Reuters, NSF, and MiniNg20).

Three significance tests have been implemented including the extended version of the micro sign test that we have derived for this study. Using these three tests, the approaches have been compared and analyzed with respect to several perspectives providing robust results. The results showed that for each extension in the methods, a corresponding significant improvement was observed in the success rates. In parallel with this result, the most advanced method which combines the leading dependencies (AWDCP) outperformed all the other methods in terms of success rates. We also observed that the optimal feature numbers showed a consistent behavior (between 2400 and 4200) in all the optimal results of the proposed methods (AWP, AWDP, and AWDCP) for all three datasets.

From the dataset perspective, an important outcome is about the formality level of the datasets. The pruning process improved the success rates of the informal MiniNg20 dataset much more than the other two formal datasets (Reuters and NSF). In addition, the formal datasets resulted in common dependencies (*adjectival modifier* and *complement*) in the leading dependency analysis, while the informal MiniNg20 had different types of dependencies (e.g. *prt* - is one of the simplest dependencies and yields the most successful results with MiniNg20).

In the next chapter, we incorporate feature selection algorithms into the proposed methods. In the experiments of the current chapter, pruning implementation was implemented for feature filtering but feature selection is different from this filtering process by using specific methods such as information gain, tf-idf, etc. These algorithms are implemented in accordance with the pruning implementation and dependency usage for text classification in the following chapter.

6. FEATURE SELECTION APPROACHES: CLASS-BASED AND CORPUS-BASED

Previously, we mostly focused on information extraction, where the main motivation was to extract new feature types (POS, dependencies, synsets, domains, etc.) in order to improve the classical bow approach. In this chapter, instead of extracting new features, we implement some kinds of information filtering where the aim is to reach and use the most critical features.

There are several feature selection algorithms which have been experimented for TC extensively. We will discuss them briefly in the following section but our aim in this chapter is not to perform another comparison with these metrics. The main motivation is to extend our current proposed system (supported by the optimal usage of dependencies and pruning levels) with an analysis of feature selection approaches, but mainly from the perspective of coverage: class-based or corpus-based coverage of features.

The chapter will begin with an overview of feature selection usage in text classification. After the examination of the overview, related experiments will be detailed.

6.1. Overview of Feature Selection Algorithms

One of the advantages of SVM over the other machine learning algorithms is its ability to deal successfully with high dimensionality [2]. In traditional studies, typically all available words in the document set were used instead of limiting to a set of keywords with satisfactory results [56, 6]. Some studies even stated that using all words leads to the best performance and using keywords may be unsuccessful without optimal parameter tuning [12, 57].

On the other hand, recent studies reveal that feature selection may improve the performance in terms of accuracy and scalability with a significant cut in the solution vector size [2]. There are different types of feature selection implementation: *Filter methods* determine a ranking among all features with respect to some statistical metrics, *wrapper methods* use classical Artificial Intelligence methods (e.g. greedy hill climbing) with cross validation, and *embedded methods* employ a linear prediction model for optimization [2]. Among them, filter methods are the simplest to implement and most scalable for text classification problems with large feature spaces [2].

There are various types of feature selection metrics for filter methods such as chi-square, information gain, tf-idf, odds ratio, probability ratio, document frequency, and bi-normal separation. Concerning these metrics, there exist many studies analyzing and comparing the metrics [12, 58], combining them based on specific measurements [59], providing unsupervised selection algorithms [60], and suggesting related novel algorithms [61].

Performing another comparison, analysis or extension of these metrics is not the main concern of this chapter, it is rather the coverage of the selected features: corpus-based (the terms that achieve the highest tf-idf score in the overall corpus are selected as the global keywords for any classification case in the dataset) and class-based (local keywords are selected for each class) feature selection approaches are analyzed with the appropriate metrics.

Concerning the studies of class-based feature selection, one of the related studies involves the categorization of internet documents [62]. A method for evaluating the importance of a term with respect to a class in the class hierarchy was proposed in that study. Another study about clustering the documents [63] focuses on increasing the speed of the clustering algorithm. For this purpose, the authors tried to make the method of extracting meaningful unit labels for document clusters much faster by using class-based keywords. In both studies, class-based feature selection approach was considered, however, the comparison was not done with all words approach or with the corpus-based feature selection approach. Another related study covers the feature selection metrics for text classification using SVM [12]. While this study makes extensive use of class-based keywords, it also does not cover an explicit comparison of the two approaches.

Experiment 1 will compare the class and corpus-based approaches by using the tf-idf metric. In the second experiment, two stage feature selection is implemented: pruning is employed as the corpus-based algorithm in addition to the class oriented use of tf-idf. Experiment 3 will be the final set-up that serializes the optimal use of dependencies (analyzed in the previous chapter) with the proposed two stage feature selection.

6.2. Experiment 1 - Introduction to Class-Based and Corpus-Based Approach in Feature Selection

In this experiment, we focus on the feature selection approach: corpus-based and class-based. Our motivation is to find an optimal keyword selection algorithm so we may describe these methods as AWK (All words with keyword selection) with several variations. In the corpus-based feature selection approach, the terms that achieve the highest tf-idf score in the overall corpus are selected as the keywords. This approach favors the prevailing classes with many documents and gives penalty to classes with small number of training documents in document corpora where there is high skew. In the class-based feature selection approach, on the other hand, distinct keywords are selected for each class. This approach gives equal weight to each class in the feature selection phase. So, less prevailing classes are not penalized as in the corpus-based approach. This approach is also suitable for the SVM classifier as it solves two class problems.

Our focus is not on the feature selection metric, any feature selection algorithm may be utilized as corpus or class-based. We prefer tf-idf, one of the simplest and commonly experimented metrics for feature selection.

6.2.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

6.2.1.1. Datasets. Reuters is used as the main dataset for the experiments.

6.2.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Porter is used for stemming implementations.
- Feature Extraction: Experiments are performed without parser (e.g. no dependency / POS information) and external knowledge base (WordNet, WND, Cyc, etc.) support.

6.2.1.3. Document Representation. Bow approach is used.

6.2.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

6.2.1.5. Machine Learning Algorithm. SVM with linear kernel is the machine learning module we use as the classification algorithm.

6.2.1.6. Feature Selection. Tf-idf is employed as the main feature selection algorithm with class-based and corpus-based perspective.

6.2.1.7. Performance Measures. Experiments are evaluated with microF and macroF scores.

6.2.2. Experiment Design

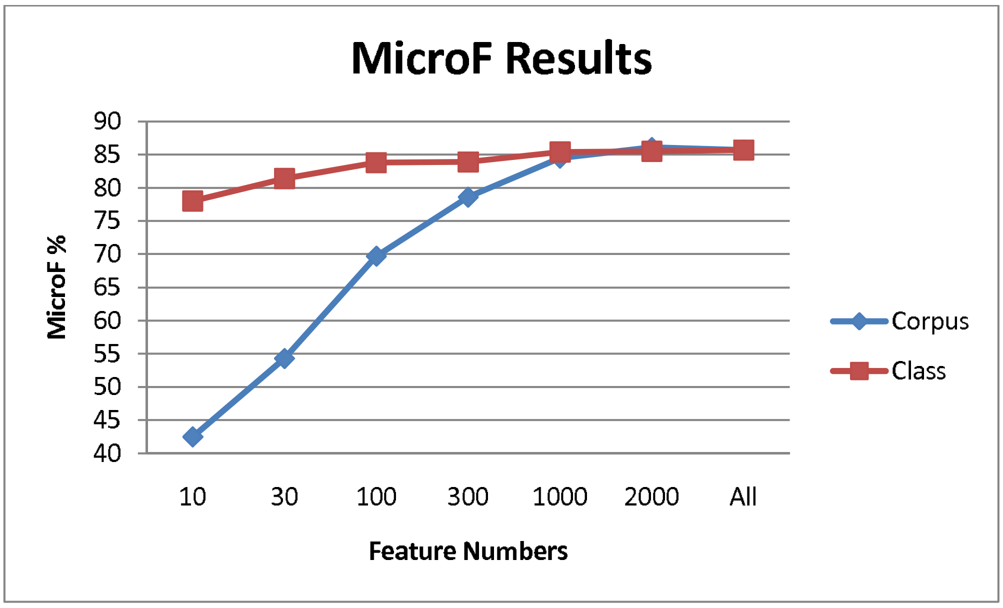
In the set-up, we experiment with six different feature number values (10, 30, 100, 300, 1000 and 2000) and perform an additional experiment containing all the words as the features for the solution vector. We repeat the same set-up for corpus-based and class-based approach, using the standard tf-idf equation.

6.2.3. Experiment Results and Comments

Figure 6.1 displays the microF and macroF results for class and corpus-based approaches with tf-idf document representations of all the words and keywords ranging in number from 10 to 2000.

For the microF results, we can conclude that class-based feature selection achieves higher microF than corpus-based approach for small number of keywords. In text classification, most of the learning takes place with a small but crucial portion of keywords for a class [64]. Class-based feature selection, by definition, focuses on this small portion; on the other hand, corpus-based approach finds general keywords concerning all classes. So, with few keywords, class-based approach achieves much more success by finding more crucial class keywords. Corpus-based approach is not successful with that small portion, but has a steeper learning curve that reaches the peak value of the experiments (86%) with 2000 corpus-based keywords.

For the macroF results, we analyze that class-based feature selection achieves consistently higher macroF performance than corpus-based approach. The high skew in the distribution of the classes in the dataset affects the macroF values in a negative way because macroF gives equal weight to each class instead of each document and documents of rare classes tend to be more misclassified. Accordingly, the average of correct classifications of classes drops dramatically for datasets having many rare classes. Class-based feature selection is observed to be very useful for this skewness. As stated above, with even a small portion of words (e.g. 100), class-based tf-idf method reaches 50% success which is far better than the 43.9% success of tf-idf with all words.



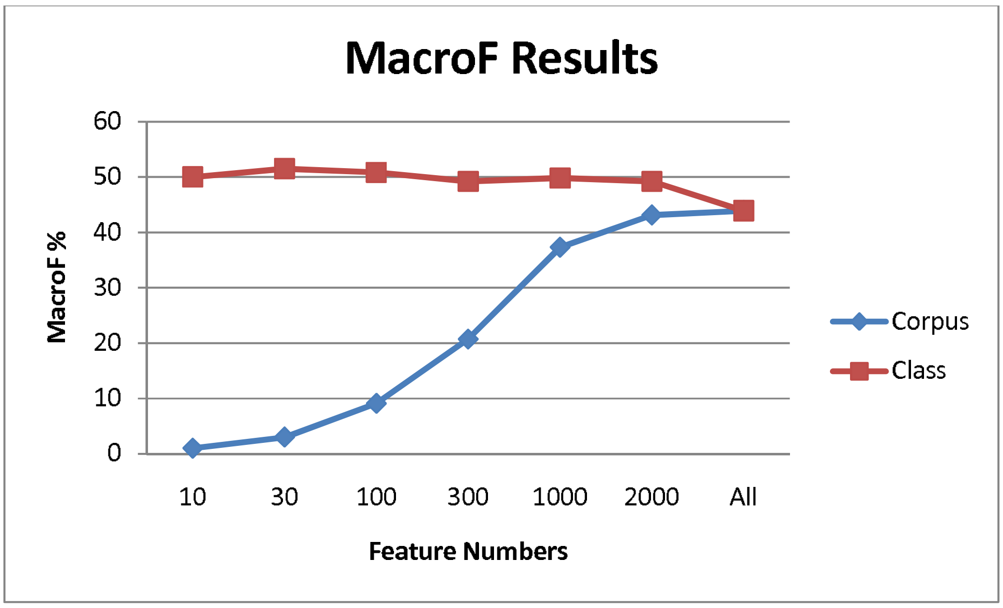


Figure 6.1. Feature selection with corpus-based and class-based approaches

Rare classes are characterized in a successful way with class-based feature selection, because every class has its own keywords for the categorization problem. The corpus-based approach shows worse results because most of the keywords are selected from prevailing classes which prevents rare classes to be represented fairly by their keywords.

6.2.4. Conclusion

In contrast to the previous studies that did not suggest feature selection in TC, we found that feature selection improves the success results of SVM. In the corpus-based approach the keywords tend to be selected from the prevailing classes. Rare classes are not represented well by these keywords. However, in the class-based approach, rare classes are represented equally well as the prevailing classes because each class is represented with its own keywords for the categorization problem. Thus, the class-based tf-idf approach with small number of keywords (e.g. 100) achieves consistently higher macroF performance than both the corpus-based approach and the approach where all the words are used. It also achieves higher microF performance than corpus-based approach when a small number of keywords is used. This is important as there is a lot of gain from classification time when a small number of keywords is used.

We can mainly conclude that the class-based approach mostly yields more successful results than the corpus-based usage. Based on this conclusion, we will use the tf-idf metric as the class-based coverage model in the following tests.

6.3. Experiment 2 - Two Stage Feature Selection

Based on the analysis of Experiment 1, we employ the tf-idf metric with the class-based approach. On the other hand, pruning implementation (which has been used as corpus-based) yielded successful results with significant improvements in the previous chapter. In this section, we want to analyze whether the incremental effect of corpus-based pruning may continue when it is combined with the class-based tf-idf metric. So, we apply two stage feature selection in this experiment: pruning is employed as the corpus-based algorithm in addition to the class oriented use of tf-idf.

The main goal is to reach features with maximum tf-idf but also with sufficient occurrence rate in the overall corpus (will be checked by the pruning levels). In this set-up, features with low occurrence rate in the overall dataset but with high frequency in a particular class (so high tf-idf) will not be included in the final feature vector of that class due to its high probability of being an outlier for the classification case.

6.3.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

6.3.1.1. Datasets. We use three datasets: Reuters, NSF and MiniNg20 [4] in order to analyze the effect of the proposed methods on different types of documents and to make comparisons between them.

6.3.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Porter is used for stemming implementations.
- Feature Extraction: Experiments are performed without parser (e.g. no dependency / POS information) and external knowledge base (WordNet, WND, Cyc, etc.) support.

6.3.1.3. Document Representation. Bow approach is used.

6.3.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

<u>6.3.1.5. Machine Learning Algorithm.</u> SVM with linear kernel is the machine learning module we use as the classification algorithm.

<u>6.3.1.6. Feature Selection.</u> Two stage feature selection algorithm is proposed. First dimension is the filtering with pruning implementation (corpus-based). Other dimension is the tf-idf filtering over each class (class-based).

<u>6.3.1.7. Performance Measures.</u> Experiments are evaluated with microF and macroF scores.

6.3.2. Experiment Design

For the life cycle of a sample experiment, features are initially filtered according to the selected PL values, and then later the ones with the sufficient PL value and the highest tf-idf values for each class are passed to the machine learning algorithm. So actually, there are two main parameters to be optimized in the experiment: *PL value* for the pruning implementation and *number of features* for the selection of the features with the highest tf-idf values.

We perform pruning level analysis using three different pruning levels: 1 (no pruning, all words are used), 2 (initial pruning level that throws away the features with single occurrence) and 13 (optimal pruning level for the words which has been analyzed in the previous experiments). For the feature number analysis, experiments are repeated with five different values: 250, 500, 1000, 2000 and 4000. So we implement a series of 15 experiments (3 different PL values * 5 different feature numbers) for each dataset.

6.3.3. Experiment Results

Experiment results for 15 different variations of the two parameters over the three datasets are listed in Table 6.1. The proposed method is named as *All words with* *keyword selection and pruning* (AWKP), the varying parameters are the pruning level (indicated by PL) and feature number (indicated by FtrNo) and they are separated by commas in the table. AWKP with PL = 1 (no pruning) is listed as AWK in the table. Benchmark results of the previous experiments (AW, AWP with PL = 2 and AWP with PL = 13) are also listed in the table in order to perform the comparison of the current set-up with the previous scores.

6.3.4. Comments and Conclusion

For the initial observation of the experiment, the parameter values were optimized in consistency with the preceding experiments: the best performance (maximum success with sufficient features) was achieved around the pruning level 13 (as in Chapter 5 during the AWP experiments) and keyword number was optimized around 2000-4000 (similar value was extracted in the previous experiment results) for all the datasets.

This optimal use of the two stage feature selection approach improved the optimal results of the previous tests significantly. Selecting the best 2000 features for each class (AWKP,13,2000) improved the previous best performance of the method AWP (with PL=13) significantly in all the three datasets as shown in Table 6.1. From the same table, we can also analyze the significant improvement of the same method (AWKP,13,2000) over the optimal method of the previous experiment without pruning (AWK,2000; class-based approach without pruning). So, briefly we have arrived at our goal in this section: we come to the conclusion that the incremental effect of corpus-based pruning continues when it is combined with the class-based tf-idf metric as the feature selection algorithm.

6.4. Experiment 3 - Two Stage Feature Selection with Dependency Usage

We now extend the previous set-up by including the optimal dependency use in all the three datasets and analyze whether the serial use of the proposed methods in this study (pruning, class-based feature selection and dependency use) improve the experimented optimal performance or not.

Table 6.1. Success scores of the two stage feature selection

	Reuters	%	NSF	%	MiniNg20	%
Method,PL,FtrNo	microF	macroF	microF	macroF	microF	macroF
AWK,250	83.69	51.15	62.04	49.51	56.65	55.72
AWK,500	84.71	50.92	62.92	49.31	56.16	55.01
AWK,1000	85.16	51.72	64.69	49.33	53.68	52.17
AWK,2000	**85.58**	**52.03**	**65.19**	**49.31**	**54.04**	**52.10**
AWK,4000	**85.84**	**52.10**	**65.71**	**49.35**	**55.25**	**53.73**
AWKP,2,250	83.99	52.49	62.11	49.96	57.19	56.06
AWKP,2,500	84.91	51.26	63.06	49.68	55.30	54.01
AWKP,2,1000	85.23	51.84	64.96	49.74	54.24	52.87
AWKP,2,2000	85.62	52.13	65.76	49.74	56.27	54.84
AWKP,2,4000	86.04	52.19	66.19	49.79	56.37	54.82
AWKP,13,250	84.56	53.35	62.88	50.38	55.47	54.78
AWKP,13,500	85.66	53.74	63.41	50.25	56.11	54.96
AWKP,13,1000	85.90	53.77	65.10	50.15	57.05	55.42
AWKP,13,2000	**86.40**	**53.95**	**66.06**	**50.11**	**57.43**	**55.66**
AWKP,13,4000	**86.70**	**53.98**	**66.10**	**50.12**	**57.43**	**55.66**
Previous Benchmarks:						
AW (AWP,1)	85.58	43.83	64.458	46.11	46.42	43.44
AWP,2	85.55	43.84	64.414	46.208	49.73	47.13
AWP,13	85.84	44.85	64.582	46.492	53.62	51.02

We apply a two stage feature selection just as the same in the previous experiment: pruning as the corpus-based algorithm and tf-idf as the class-based metric. We extend the solution vector with the optimal use of dependencies for each dataset that have been previously analyzed by the AWDP approach in Chapter 5. So, the current motivation in this section is to perform a detailed analysis where we study combining possible feature selection metrics with pruning for both word and dependency features.

6.4.1. Experiment Configuration According to TC Fundamentals

Below sections discuss the main preferences of the proposed experiment according to the fundamentals of text classification described in Chapter 2.

6.4.1.1. Datasets. We use three datasets: Reuters, NSF and MiniNg20 [4] in order to analyze the effect of the proposed methods on different types of documents and to make comparisons between them.

6.4.1.2. Preprocessing Operations. Preprocessing is performed with the below routines.

- Feature Formatting: Non-alphabetic characters and mark-up tags are discarded, case-folding and removal of stopwords are performed. Porter is used for stemming implementations.
- Feature Extraction: We employ Stanford Parser for the extraction of dependencies. Experiments are performed without external knowledge base (WordNet, WND, Cyc, etc.) support.

6.4.1.3. Document Representation. Bow approach is used.

6.4.1.4. Term Weighting Approach. We choose the standard tf-idf metric for term weighting in our methods.

6.4.1.5. Machine Learning Algorithm. SVM with linear kernel is the machine learning module we use as the classification algorithm.

6.4.1.6. Feature Selection. Two stage feature selection algorithm is proposed. First dimension is the filtering with pruning implementation for both types of features (words and dependencies). Other dimension is the tf-idf filtering over each class.

6.4.1.7. Performance Measures. Experiments are evaluated with microF and macroF scores. Micro sign test, macro sign test and micro sign test with positive instances are performed to measure the significance of the improvement.

6.4.2. Experiment Design

In this experiment, we serialize the use of pruning, dependencies and class-based feature selection in text classification. For the set-up, two types of features employed: words and dependencies. Both feature types are initially filtered according to their corresponding optimal PL values, and then later the features with the sufficient PL value and the highest tf-idf values for each class are passed to the machine learning algorithm.

For the parameters of the experiment (PL values for dependencies and words, keyword number), we prefer the values around the optimal values analyzed in the previous experiments. 2000 and 4000 keywords are analyzed for all the three datasets. We use 13 as the PL value for words in all the datasets. For dependencies; 50, 8 and 8 are taken as the optimal dependency PL values for Reuters, NSF, and MiniNg20 respectively.

6.4.3. Experiment Results

The proposed method of this section is named as *All words and optimal dependency combinations with keyword selection and pruning* (AWDCKP). We have designed

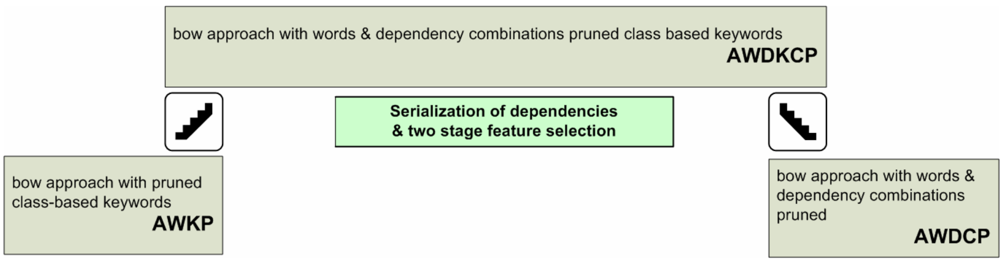

bow approach with words & dependency combinations pruned class based keywords
AWDKCP
Serialization of dependencies & two stage feature selection
bow approach with pruned class-based keywords
AWKP
bow approach with words & dependency combinations pruned
AWDCP

Figure 6.2. Inheritance of AWDCKP from AWKP and AWDCP

Table 6.2. Success scores of the dependency support with two stage feature selection

	Reuters	%	NSF	%	MiniNg20	%
Method, Params	**microF**	**macroF**	**microF**	**macroF**	**microF**	**macroF**
AWDCKP,13,opt,2000	86.38	52.65	**67.02**	**50.93**	**58.10**	**56.26**
AWDCKP,13,opt,4000	86.54	52.68	**67.23**	**50.94**	**58.10**	**56.26**
Previous Benchmarks						
AWKP,13,2000	**86.40**	**53.95**	66.06	50.11	57.43	55.66
AWKP,13,4000	**86.70**	**53.98**	66.10	50.12	57.43	55.66
AWDCP, opt	86.03	45.26	66.01	47.68	54.23	51.65

the experiment varying with only keyword number (2000 and 4000) for AWDCKP, taking the other parameters (PL value for words and dependencies) as their optimal values referencing the previous optimal results. The results are listed in Table 6.2. AWDCKP inherits its discriminative properties and parameter values directly from the previous two methods: AWDCP and AWKP as shown in Figure 6.2. So, the benchmark results of the optimal set-up of these previous experiments are also listed in the table in order to perform the comparison of the current set-up with the previous most successful scores.

The parameters of the methods are separated by commas, following the method name. For example, *AWDCP, opt* is the optimal AWDCP method analyzed in Section 5.4.1. On the other hand, *AWKP,13,4000* is the AWKP experiment with PL = 13 for words and 4000 keywords. *AWDCKP,13,opt,2000* combines the use of these two methods with the analyzed optimal values (e.g. 2000 keywords, word PL = 13, etc).

Table 6.3. Statistical comparison of the proposed methods - AW, AWK, AWKP and AWDCKP

	Micro Sign, All	Micro Sign, +	Macro Sign
Reuters			
AWK over AW	$\sim$	$\gg$	$\gg$
AWKP over AWK	$\gg$	$\gg$	$>$
AWKP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AWK	$\gg$	$\gg$	$\sim$
AWDCKP over AWKP	$\sim$	$<$	$\sim$
NSF			
AWK over AW	$\gg$	$\gg$	$\gg$
AWKP over AWK	$\sim$	$\gg$	$>$
AWKP over AW	$\sim$	$\gg$	$\gg$
AWDCP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AWK	$\gg$	$\gg$	$\gg$
AWDCKP over AWKP	$\gg$	$\gg$	$\gg$
MiniNg20			
AWK over AW	$\gg$	$\gg$	$\gg$
AWKP over AWK	$\sim$	$>$	$>$
AWKP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AWK	$\sim$	$>$	$\sim$
AWDCKP over AWKP	$>$	$\sim$	$>$
All datasets			
AWK over AW	$\gg$	$\gg$	$\gg$
AWKP over AWK	$\sim$	$\gg$	$\gg$
AWKP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AW	$\gg$	$\gg$	$\gg$
AWDCKP over AWK	$\gg$	$\gg$	$\gg$
AWDCKP over AWKP	$\gg$	$\gg$	$\gg$

6.4.4. Significance of the Improvements

Table 6.3 compares the methods proposed in this chapter and the standard benchmark method, AW. Three significance tests that have been defined in Section 2.7.2 (and previously used in Chapter 5) were applied. The methods were compared according to the z values with the corresponding confidence areas in z-table. Repeating the same significance test set-up with the same terminology, we list the results of the comparisons in Table 6.3.

It is observed from the table that each method significantly outperforms its predecessor method (AWDCKP >> AWKP >> AWK >> AW) and AWDCKP in most of the datasets and in the overall case with the three significance measures. So the most advanced proposed method (AWDCKP) is significantly the optimal method. There is only one exception - according to the micro sign test with positive instances in Reuters, AWDCKP is observed to be significantly unsuccessful than AWKP. On the other hand, the other two significance tests claim that both methods are similarly successful in Reuters.

6.4.5. Comments and Conclusion

We can briefly say that AWDCKP yields generally the best results among all the discussed algorithms. The proposed method is more successful than AWDCP in all datasets and this superiority continues over AWKP in NSF and MiniNg20. The only exception is in the Reuters dataset where AWKP is the most successful method. AWDCKP yields similar results according to two significance measures but it is worse than AWKP with respect to the micro sign test with positive instances. AWKP, having the two stage feature selection approach, directly employs bow approach without dependency use. Reuters differs from the other two datasets by having more and longer documents with much more content; its high formality level also improves the positive effect of the words. So we may conclude that for a formal dataset with sufficient word coverage in the solution (by employing the optimal feature selection approach), we may not need dependency support for the optimal classification case. On the other hand, NSF with short journal abstracts and informal MiniNg20 benefit from the dependency support as well as the two stage feature selection.

7. CONCLUSIONS

In this thesis, preferring the state-of-the-art methods in the other stages as much as possible, we attack the text classification problem from mainly two fundamentals: feature extraction and feature selection. Feature extraction mainly focuses on extracting new and informative feature types which are mostly syntactic or semantic oriented. Main motivation of this process is to extend and improve the solution vector (bow approach with only morphological concern) by adding the missing points of this standard approach. On the other hand, feature selection tries to filter uninformative features with some statistical formulations in order to yield more scalable and accurate solutions.

In the introductive parts of the study, utilization of external knowledge bases for informative semantic features was the initial motivation and Cyc seemed to be one of the key ontologies for that purpose with its stated power in the text domain. But, the practical implementation of Cyc in the text classification domain was far away from being satisfactory. The ontology lacked critical points such as syntactic and semantic analysis at sentence level. In addition, our implementation trials suffered from big time complexities of the function calls. Due to these problems, we have not continued the experiments with Cyc and decided to use another popular lexical knowledge base, WordNet. For the WordNet and its related tool WND use, synsets were analyzed to be more powerful than WND domains in achieving successful classification results but the overall results were not satisfactory with respect to the benchmark results of the bow approach due to the ambiguity problem. The mentioned ambiguity problem was also mentioned in parallel studies with any other ontologies and knowledge bases so we suggest that the conceptual contribution of ontologies is not an effective way yet to increase classification accuracy in the document datasets.

For the morphological part, one main contribution was the approach in stemmer utilization. Stemming was analyzed not only for the words but also for all the extracted dependency couples in the texts. Porter stemming was observed to be the optimal stemmer for all words while the raw form without stemming slightly outperformed the other approaches in dependency stemming.

For the syntactic part, dependency based methods have shown better performance than POS based methods which was a promising and original result for the future direction of the study at that time. The most successful dependency types were improving the benchmark by around 0.4%-0.5%. One of the main observations of the experiments was that most of the dependency types yielded many instances that caused highly sparse solution vectors. This high sparsity indicated a feature selection implementation to be an explicit need for the proposed system. So at that point, we decided to deal with pruning analysis of features as the basic feature selection algorithm and perform further analysis of dependencies (e.g. combination of the leading ones).

After the completion and the corresponding analysis about the preliminary experiments, the main research directions and possible contributions were mainly shaped. We performed an analysis for the optimal pruning of the features (10 different levels were tested for pruning implementation). Pruning was implemented incrementally in three main stages: pruning of words, pruning of dependencies, and pruning of dependency combinations. We experimented with 36 lexical dependencies independently with the optimal pruning levels and performed the final test by using the combination of the leading dependencies in addition to all the words in the documents. Later, we compared the performance of corpus-based and class-based approaches for feature selection coverage and then extended the corpus-based pruning implementation by the tf-idf class-based feature selection. We examined a range between 10 and 4000 for keyword numbers and achieved the optimal parameters for the experimented datasets. For the final test, we serialized the optimal use of the leading dependencies for each experimented dataset with the two stage feature selection approach.

By performing pruning level analysis on datasets with different characteristics and obtaining the optimal results around PL=13 consistently, we arrive at the conclusion that a pruning level of about 10-15 (removing words occurring less than 10-15 times) may give the best results for a dataset. In the literature, usually an arbitrarily selected and small value (e.g. PL=2 or PL=3) has been used for this purpose.

We also observed that the optimal feature numbers showed a consistent behavior (around 2000 and 4000) in all the optimal results of the proposed methods for all three datasets. From the dataset perspective, an important outcome is about the formality level of the datasets. The proposed algorithms (pruning, class-based selection, dependency use) generally improved the success rates of the informal MiniNg20 dataset much more than the other two formal datasets (Reuters and NSF). In addition, the formal datasets resulted in common dependencies (*adjectival modifier* and *complement*) in the leading dependency analysis. However, in the informal MiniNg20 dataset, *comp* does not improve the performance of the classifier due to the simple and ungrammatical sentence structures in the dataset. Instead of it, simple *prt* (phrasal verb participle, e.g. *write down*) dependency yields the most successful results with this informal dataset. Another important analysis is about the massive improvement of macroF with Reuters data when the class-based feature selection is implemented. Class-based feature selection compensated the high skewness of the dataset by giving equal weight to each class in the feature selection phase. So, less prevailing classes were not penalized which caused the improvement in macroF.

Three significance tests have been implemented to test for the robustness of the results and the significance of the improvements. Besides the classical micro and macro sign tests, we derived an extended version of the micro sign test for this study. The results showed that for each extension in the methods, a corresponding significant improvement was observed in the success rates. Figure 7.1 shows a general system topology of all the successfully proposed approaches in the thesis. Each proposed method is shown with its discriminative properties and the link (stair) is supplied to its inherited / related method.

The figure also reflects the improvements for each proposed approach: the stairs show the inheritances between the methods and the size of it roughly indicates the significance of the improvement. The improvement rates (so the size of the stairs in the figure) may change with respect to:

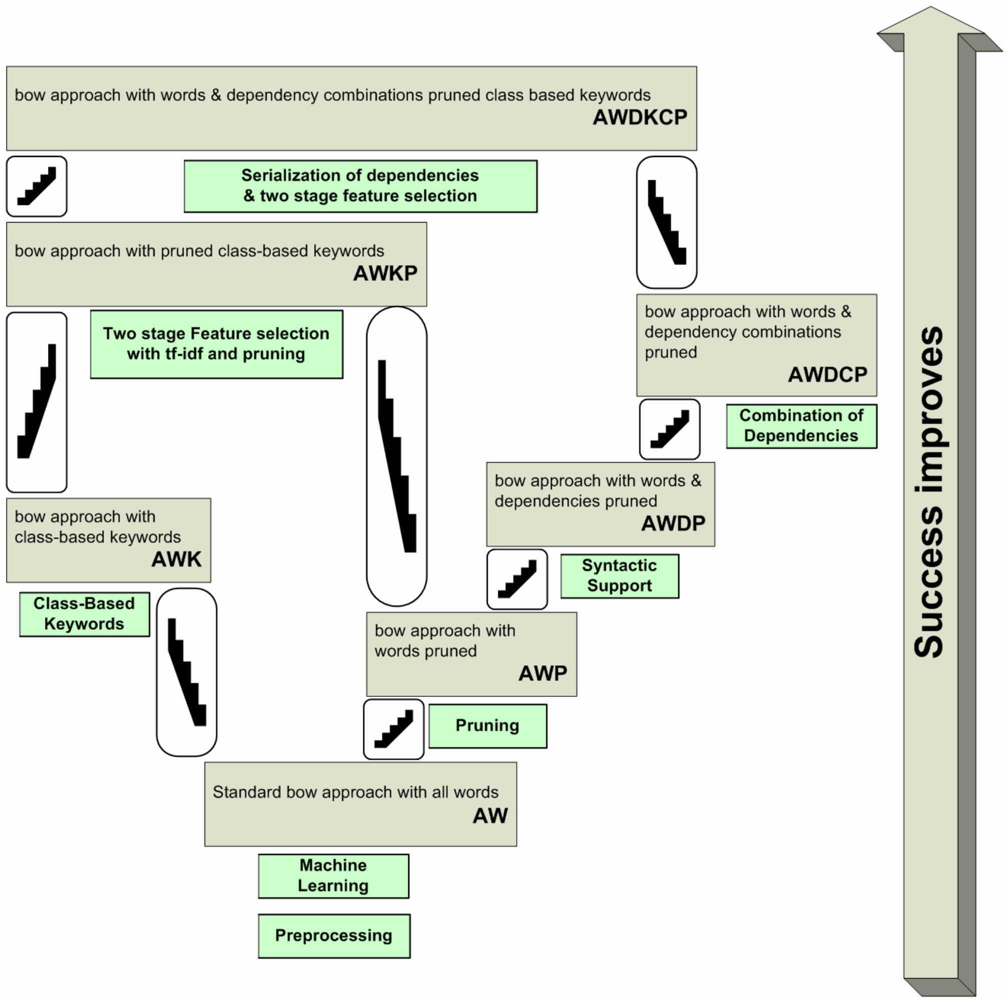

bow approach with words & dependency combinations pruned class based keywords
AWDKCP
Serialization of dependencies & two stage feature selection
bow approach with pruned class-based keywords
AWKP
bow approach with words & dependency combinations pruned
AWDCP
Two stage Feature selection with tf-idf and pruning
Combination of Dependencies
bow approach with words & dependencies pruned
AWDP
bow approach with class-based keywords
AWK
Syntactic Support
Class-Based Keywords
bow approach with words pruned
AWP
Pruning
Standard bow approach with all words
AW
Machine Learning
Preprocessing
Success improves

Figure 7.1. Overview of the improvements of the proposed approaches

- Success measure: Class-based methods are usually better with macroF, but microF performance is not so high.
- Dataset properties: Reuters is highly skewed so its macroF performance is much more affected by Class-based keywords. For the MiniNg20 case, the bow representation was unsuccessful due to the informal content of the dataset with lots of textual errors. So any proposed algorithm (e.g. pruning, dependency use) caused much more significant improvement in this dataset by compensating the insufficient representation of bow. A total of about 25% increase (from about 45% to 57% in both microF and macroF) was achieved in MiniNg20 from AW to AWKPCP.

The most advanced method combining the leading dependencies with optimal pruning levels and optimal number of class-based features (AWDCKP) mostly outperforms the other methods in terms of success rates with reasonable feature sizes. The proposed method is more successful than AWDCP in all the datasets and this superiority continues over AWKP in NSF and MiniNg20. The only exception is Reuters dataset where AWKP is the leading method. Although it yields similar results according to two significance measure, AWDCKP is worse than AWKP with respect to the micro sign test with positive instances in Reuters. This dataset differs from the other experimented ones by having more and longer documents with much more content that increases the effect of the bow approach with more consistent word occurrences; its high formality level also improves this effect. So, for a formal dataset with sufficient content in the documents, we may not need dependency support for the optimal classification result in case of the selection of optimal features per class, as in the case of AWKP. On the other hand, confirming the superiority of the AWDCKP method, NSF with short journal abstracts and the informal MiniNg20 benefit from the dependency support as well as the two stage feature selection for classifying their documents.

To the best of our knowledge, this is the first study that makes such a detailed analysis on individual dependencies and feature selection coverage with a two stage selection in not only text classification but more generally in the text domain. For future work, remembering the analogy of text classification with the *elephant among* *blind researchers* concept [2], there are many possible research directions related to this problem with several solution strategies. One possible projection is to repeat the implementation of the proposed successful methods in more datasets with different formality levels, document lengths, and skewness properties so that we can develop robust algorithms for automatic detection of possible useful dependencies and corresponding pruning levels / feature selection approaches according to the dataset properties. Another possible extension may be the employment of the proposed approaches in other document oriented problems. Using the two stage feature filtering or suggesting a novel index type of the leading lexical dependencies in search engines or file organization systems may probably improve the related performance (e.g. ranking with respect to relevance in search, or success of retrieving the required document) on that system. From another perspective, any text classification study preferring the proposed approaches of this thesis and focusing on the other fundamentals (e.g. improvements of the SVM implementation, improving the linear kernel, successful use WordNet, etc.) will be a related novel study in this domain.

REFERENCES

1. Alpaydın, E., *Introduction to Machine Learning*, The MIT Press, 2004.

2. Liu, H. and M. Hiroshi, *Computational Methods of Feature Selection*, Chapman and Hall/CRC Press, HPL-2007-16, 2007.

3. Manning, C.D., P. Raghavan and H. Schütze , *Introduction to Information Retrieval*, Cambridge University Press, 2008.

4. Asuncion, A. and D. Newman, UCI Machine Learning Repository, *Irvine, CA: University of California, School of Information and Computer Science*, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2007.

5. Özgür, A., L. Özgür, and T. Güngör, "Text Categorization with Class-Based and Corpus-Based Keyword Selection", *Proceedings of ISCIS 2005, Lecture Notes in Computer Science,* Vol.3733, pp. 606-615, Springer-Verlag, Berlin Heidelberg, 2005.

6. Yang, Y. and X. Liu, "A Re-examination of Text Categorization Methods", *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval,* Berkeley, 1999.

7. Salton, G., C.S. Yang, A. Wong, "A Vector-Space Model for Automatic Indexing", *Communications of the ACM 18* no.11, 1975.

8. Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Christianini and C. Watkins, "Text Classification using String Kernels", *Journal of Machine Learning Research,* pages 419-444, 2002.

9. Salton, G. and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management 24*, no. 5, pp. 513–523, 1988.

10. Robertson, S.E., S. Walker and M. Beaulieu, "Experimentation as a way of life:

Okapi at TREC", *Information Processing and Management 36*, pp. 95-108, 2000.

11. Joachims, T., "Advances in Kernel Methods-Support Vector Learning. Making Large-Scale SVM Learning Practical", MIT-Press, 1999.

12. Forman, G., "An Extensive Empirical Study of Feature Selection Metrics for Text Classification", *Journal of Machine Learning Research 3*, pp. 1289–1305, 2003.

13. Burges, C. J. C., "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 121–167, 1998.

14. Wang, Z. and D. Zhang, "Feature Selection in Text Classification Via SVM and LSI", *ISNN 2006, Lecture Notes in Computer Science,* Vol. 3971, pp. 1381–1386, Springer-Verlag, 2006.

15. Liu, T., Z. Chen, B. Zhang, W. Ma and G. Wu, "Improving Text Classification using Local Latent Semantic Indexing", *Proceedings of the Fourth IEEE International Conference on Data Mining* (ICDM–2004), 2004.

16. Larson, R. and B. Farber, *Elementary Statistics : Picturing the World*, Prentice Hall, 2000.

17. Montgomery, D. C., *Design and Analysis of Experiments*, John Wiley, 2001

18. Stevenson, M. and M. Greenwood, "A Semantic Approach to IE Pattern Induction", *Proceedings of the 43rd Annual Meeting of the ACL*, Ann Arbor, 2005.

19. Stevenson, M. and M. Greenwood, "Comparing Information Extraction Pattern Models", *Proceedings of the Workshop on Information Extraction Beyond the Document,* pp. 12-19, Sydney, 2006.

20. Marneffe, M.C., B. MacCartney and C. Manning, "Generating Typed Dependency Parses From Phrase Structure Parses", *LREC2006,* 2006.

21. Klein, D. and C. Manning, “Fast Exact Inference with a Factored Model for Natural Language Parsing”, *NIPS,* volume 15, MIT Press, 2003.

22. Levy, R. and G. Andrew, “Tregex and Tsurgeon: tools for querying and manipulating tree data structures”, *5th International Conference on Language Resources and Evaluation (LREC 2006)*, 2006.

23. Wong, A.K.S., J.W.T. Lee and D.S. Yeung, “Using complex linguistic features in context-sensitive text classification techniques”,*Proceedings of Machine Learning and Cybernetics*, Volume 5, Issue, 18-21 Aug. 2005, pp. 3183 - 3188 Vol. 5, 2005.

24. Madsen, R. E., J. Larsen and L.K. Hansen, “Part-of-Speech Enhanced Context Recognition”, *Proceedings of IEEE Workshop on Machine Learning for Signal Processing XIV*, pp. 635-644, IEEE Press, 2004.

25. Goncalves, T., C. Silva, P. Quaresma and R. Vieira, “Analysing Part-of-Speech for Portuguese Text Classification”, *CICLing 2006*, pp. 551-562, 2006.

26. Mullen, T. and N. Collier, “Sentiment analysis using support vector machines with diverse information sources”, *Proceedings of EMNLP 2004*, 2004.

27. Cahill, A., U. Heid, C. Rohrer and M. Weller, “Using tri-lexical dependencies in LFG parse disambiguation”, *The 14th International LFG Conference*, July 2009, Trinity College, Cambridge, United Kingdom, 2009.

28. Charniak, E., K. Knight and K. Yamada, “Syntax-based language models for statistical machine translation”, *Proceedings of the MT Summit IX. International Association for Machine Translation*, 2003.

29. Herrera, J., A. Penas, and F. Verdejo, “Textual Entailment Recognition Based on Dependency Analysis and WordNet”, *Lecture Notes in Computer Science*, Volume 3944/2006, Springer Berlin / Heidelberg, 2006.

30. Wellner, B., J.D. Pustejovsky, C. Havasi, A. Rumshisky and R. Sauri, “Classifi-

cation of Discourse Coherence Relations: An Exploratory Study using Multiple Knowledge Sources", *Proceedings of the SIGdial Workshop On Discourse And Dialogue*, 2006.

31. Basili, R., M.T. Pazienza and L. Mazzucchelli, "An Adaptive and Distributed Framework for Advanced IR", *RIAO 2000*, pp. 908-922, 2000.

32. Lewis, D. D., "An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task", *Proceedings of SIGIR-92,* pp. 37-50, Copenhagen, Denmark, 1992.

33. Furnkranz, J., T. Mitchell and E. Rilof, "A Case Study in Using Linguistic Phrases for Text Categorization on the WWW", *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

34. König, A.C. and E. Brill, "Reducing the Human Overhead in Text Categorization", *Proceedings of KDD 2006*, Association for Computing Machinery Inc., 2006.

35. Moschitti, A. and R. Basili, "Complex Linguistic Features for Text Classification", *A Comprehensive Study, ECIR 2004*, pp. 181-196, 2004.

36. Moschitti, A., "Kernel Methods, Syntax and Semantics for Relational Text Categorization", *Proceeding of ACM 17th Conference on Information and Knowledge Management (CIKM)*, Napa Valley, California, 2008.

37. Ghanem, M., Y. Guo, H. Lodhi and Y. Zhang, "Automatic Scientific Text Classification using Local Patterns: KDD CUP 2002 (Task1)", *SIGKDD Explorations,* vol. 4, no. 2, pp. 95-96, 2002.

38. Nastase, V., J.S. Shirabad and M.F. Caropreso, "Using Dependency Relations for Text Classification", *AI 2006, the nineteenth Canadian Conference on Artificial Intelligence,* Québec City, Quebec, Canada, 2006.

39. Özgür, L. and T. Güngör, "Analysis of Stemming Alternatives and Dependency

Pattern Support in Text Classification", *CICLing 2009, the tenth International Conference on Intelligent Text Processing and Computational Linguistics,* Mexico City, Mexico. Research in Computing Science, 41, 2009.

40. Gruber, T., "A Translation Approach to Portable Ontologies", *Knowledge Acquisition*, 5(2), pp. 199,220, 1993.

41. Magnini, B. and G. Cavaglia, "Integrating Subject Field Codes into WordNet", *Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation,* Athens, Greece, 31 May - 2 June, 2000, pp. 1413-1418, 2000.

42. Bentivogli, L., P. Forner, B. Magnini and E. Pianta, "Revising WordNet Domains Hierarchy: Semantics, Coverage, and Balancing", *Proceedings of COLING 2004 Workshop on Multilingual Linguistic Resources,* Geneva, Switzerland, August 28, 2004, pp. 101-108, 2004.

43. Kozareva, Z., S. Vazquez S. and A. Montoyo, "The Usefulness of Conceptual Representation for the Identification of Semantic Variability Expressions", *CICLing 2007*, pp. 325-336, 2007.

44. Koeling, R., D. McCarthy and J. Carroll, "Text categorization for improved priors of word meaning", *Proceedings of the Eighth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2007)*, Mexico City, Mexico, 3rd Best Paper Award, 2007.

45. Strapparava, C. and A. Valitutti, "WordNet-Affect: an affective extension of WordNet", *Proceedings of 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pp. 1083–1086, Lisbon, May 2004.

46. Ding, Y., "Ontology: The enabler for the Semantic Web", Technical Report, Division of Mathematics and Computer Science, Free University, Amsterdam, 2001.

47. Cycorp Inc., *Cycorp Cyc ontology homepage*, http://www.cyc.com, 2010.

48. Wikipedia, *The Free Encyclopedia*, http://en.wikipedia.org/wiki/Wikipedia, 2010.

49. Open Mind, *MIT Media Lab Project - Open Mind Common Sense homepage*, http://openmind.media.mit.edu, 2010.

50. Liu, H. and P. Singh, "ConceptNet - a practical commonsense reasoning tool-kit", *BT Technology Journal*, Vol 22 No 4, 2004.

51. Hsu, M.H. and H. Chen, "Information retrieval with commonsense knowledge", *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006.

52. Miller, A., "WordNet: Lexical Database for English", *Communications of the ACM*, Vol. 38, pp: 39,41, 1995.

53. Hidalgo, J.M.G and M.B. Rodriguez, "Integrating a Lexical Database and a Training Collection for Text Categorization", *ACL/EACL Workshop on Automatic Extraction and Building of Lexical Semantic Resources for Natural Language Applications*, 1997.

54. Mansuy, T. and R. Hilderman, "A Characterization of WordNet Features in Boolean Models for Text Classification", *Proceedings of the 5th Australasian Data Mining Conference (AusDM'06)*, Sydney, Australia, November, pp. 103-109, 2006.

55. Porter, M., "An Algorithm for Suffix Stripping", *Program 14*, pp. 130-137, 1980.

56. Joachims, T., "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", *European Conference on Machine Learning (ECML)*, pp. 137-142, Berlin, Springer, 1998.

57. Aizawa, A., "Linguistic Techniques to Improve the Performance of Automatic Text Categorization", *Proceedings of 6th Natural Language Processing Pacific Rim Sym-*

posium, Tokyo, JP, pp. 307-314, 2001.

58. Yang, Y. and J. O. Pedersen, “A Comparative Study on Feature Selection in Text Categorization”, *Proceedings of the 14th International Conference on Machine Learning*, pp. 412–420, 1997.

59. Shoushan, L., X. Rui, Z. Chengqing and C.R. Huang, “A Framework of Feature Selection Methods for Text Categorization”, *Proceedings of the 47th Annual Meeting of the ACL and the 4t IJCNLP of the AFNLP*, pp. 692-700, Suntec, Singapore, 2009.

60. Dasgupta, A., P. Drineas, B. Harb, V. Josifovski and M.W. Mahoney, “Feature Selection Methods for Text Classification”, *Proceedings of 13th Annual SIGKDD*, pp. 230-239, 2007.

61. Shang, W., H. Huang, H. Zhu, Y. Lin, Y. Qu and W. Zhihai, “A Novel Feature Selection Algorithm For Text Categorization”, *Expert Systems with Applications,* Vol. 33, pp. 1-5, 2007.

62. Lin, S-H., C-S. Shih, M. C. Chen and J-M Ho, “Extracting Classification Knowledge of Internet Documents with Mining Term Associations: A Semantic Approach”, *Proceedings of ACM/SIGIR,* Melbourne, Australia 241–249, 1998.

63. Azcarraga, A. P., T. Yap, and T. S. Chua, “Comparing Keyword Extraction Techniques for Websom Text Archives”, *International Journal of Artificial Intelligence Tools*, 11 no. 2, 2002.

64. Özgür, L., T. Güngör and F. Gürgen, “Adaptive Anti-Spam Filtering for Agglutinative Languages. A Special Case for Turkish”, *Pattern Recognition Letters,* **25** no.16 pp. 1819–1831, 2004.