Analysis of Stemming Alternatives and Dependency Pattern Support in Text Classification

Levent Özgür and Tunga Güngör

Department of Computer Engineering, Boğaziçi University,
Bebek, 34342 Istanbul,Turkey
{ozgurlev,gungort}@boun.edu.tr

Abstract. In this paper, we study text classification algorithms by utilizing two concepts from Information Extraction discipline; dependency patterns and stemmer analysis. To the best of our knowledge, this is the first study to fully explore all possible dependency patterns during the formation of the solution vector in the Text Categorization problem. The benchmark of the classical approach in text classification is improved by the proposed method of pattern utilization. The test results show that support of four patterns achieves the highest ranks, namely, *participle modifier*, *adverbal clause modifier*, *conjunctive* and *possession modifier.* For the stemming process, we benefit from both morphological and syntactic stemming tools, Porter stemmer and Stanford Stemmer, respectively. One of the main contributions of this paper is its approach in stemmer utilization. Stemming is performed not only for the words but also for all the extracted pattern 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 pattern stemming. For the implementation of our algorithm, two formal datasets, Reuters - 21578 and National Science Foundation Abstracts, are used.

Key words: Text Classification, Dependency Patterns, Stemmer Analysis, Information Extraction

1 Introduction

Text Classification (TC) is a learning task, where pre-defined category labels are assigned to documents based on the likelihood suggested by a training set of labelled documents.

Most of the approaches used in this problem study it in bag-of-words (bow) form, where only the words in the text are analyzed by some machine learning algorithms for TC [1]. In this approach, documents are represented by the widely used vector-space model, introduced by Salton et al. [2]. In this model, each document is represented as a vector d. Each dimension in the vector d stands for a distinct term (word) in the term space of the document collection.

Received 17/11/08
Accepted 15/12/08
Final version 03/02/09

Classical bow approach is solely based on the words of the sentence without any further study on the implicit concepts behind them. Although the success scores are found to be satisfactory in most studies, this approach may not be so adequate for some complex cases. Dealing with semantic similarity and concepts is a critical and challenging subject in TC.

WordNet is the most experienced lexical tool in text related studies [3]. Briefly, WordNet introduces *synset* concept which corresponds to *synonym set*. Basic studies utilizing WordNet in TC have not yielded outstanding results because of mainly the disambiguation problem [4] [5]. There are also positive results but with specific exceptions, like manual disambiguation [6]. On the other hand, recent boosting algorithms have yielded successful results [7]. Also supplementary packages of WordNet (i.e: QueryData, Similarity Package) have been utilized recently which have been stated to improve the performance but by also increasing the complexity of the solution [8].

Almost all of these approaches lack the fact that the meaning of a sentence may not always be explicitly presented within the words; the sentence may contain implicit facts that can only be sensed through a deep analysis of the whole sentence by examining its syntactic and semantic structure. In order to fill this gap, we benefit from Information Extraction (IE) discipline, which aims to extract structured information from unstructured machine-readable documents. Patterns and sentence dependencies are recent topics in this discipline which we analyze in this study. To make an exploration of this new possible feature, we use 22 different pattern types and enrich our solution vector separately with these new features.

Stemmer analysis in text processing is our other major concern in this paper. In almost all previous studies, we see that morphological stemming, in which stemming is based on only morphological issues that are completely independent from the syntactic and semantic structure of the sentence, is always performed as a standard preprocess operation while forming the solution vector. Both inflections and derivational affixes are removed in this type. Utilizing the stemmed form of the word instead of its raw form may be preferred in straightforward approaches (i.e. bow approach), but our perspective in this paper introduces the dependency couple of words which increases the occurrence of several words, so a reexamination of stemming algorithms in this integrated approach will be the contribution of this paper. Moreover, the effectiveness of the straightforward style of morphological stemming should also be questioned systematically. In this paper, in addition to the morphological stemmers, we also analyze syntactic stemmers. We emphasize the name of this alternative type as syntactic stemming because this type of stemming is performed during the syntactic analysis of the sentence in which POS information and lexical dependencies are analyzed. Different from the morphological parsing, only inflections are removed by keeping the derivational affixes in this type of stemming. For example, the word *arrivals* is stemmed as *arrive* in Porter stemmer while the base form is found as *arrival* in Stanford stemmer by keeping the derivational affix. We utilize both ways of stemming by employing two well-known tools: Porter Stemmer [9] as the morphological stemmer and Stanford Stemmer, the built-in stemmer of Stanford Parser, as the syntactic one. In this paper, we also count the WordNet synset utilization as another alternative stemming style because WordNet usage is also highly related with stemming. Porter stemmer is not compatible with WordNet because the POS information of the raw word is lost due to the removal of the derivational affixes. Stanford stemming is implemented before WordNet utilization and the synset id, which is extracted by WordNet after stemming, is also stated as another stemming type.

The paper is organized as follows: Details of the pattern concept are discussed in Section 2. Our proposed model is covered in Section 3 and test results are given in Section 4. We conclude the paper and propose future work in Section 5.

2 Dependency Patterns

2.1 Related Studies Based on Patterns

A critical problem in IE is to develop systems which can be easily adapted to new domains as automatically and correctly as possible [10]. Solutions to this problem attempt to learn the domain-specific information, named as patterns. Patterns can be structured in many different ways with different levels of linguistic analysis. In a detailed analysis between different pattern structures [11], four different pattern models were analyzed which are predicate-argument model (SVO), chains, linked chains and subtrees. Riloff devised an original algorithm in her remarkable study, which automatically generates significant extraction patterns with noun generalization from untagged texts [12].

Lexical dependency is a different way of representing the structure of the sentences which extracts grammatical relations (*object*, *subject*, *preposition* etc.) between words in a sentence [13]. An increasing interest in using lexical dependency properties for different NLP tasks from machine translation to question answering is observed in the related studies. A recent study has focused on dependency support for text classification which has yielded successful results but with a narrow view utilizing all the dependencies together without a further and detailed analysis [14]. To make an exploration of this new possible feature, we perform a further analysis by studying each pattern specifically as an extension of the standard bow approach which is explained in Section 2.2.

Parser utilization is inevitable for syntactic study in which the phrases, POS information and dependency of the words in the sentences are identified. Our implementation details for this subject are given in Section 3.1.

2.2 Dependency Pattern Utilization

22 grammatical relations are employed in the tests from the list of 48 relations given in [13]. Table 1 shows these relations; including their definitions and some examples. Our selection criterion is highly motivated by their utilization frequencies and also their generalization capacities. Dependencies are eliminated

including number contents (i.e. *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.

Table 1. Dependency Patterns and Their Examples

Symbol	Pattern Type	Example Couples	Symbol	Pattern 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

3 Proposed Model

3.1 Modules

Syntactic Tool Stanford Parser is known to be the most powerful and efficient parser in the subject. In our tests with this parser, the parser is observed to avert syntactic ambiguities. In the most recent version, its structure gives only the first probable parse as the result. It is compared with two other systems and rated as the parser with the least error rate [11]. It has an integrated capability of extracting both the POS information and the dependencies between the words in a sentence. PCFG parser mode is selected in our implementation which extracts this information directly instead of a factorized solution [15].

Lexical Tool WordNet, as explained above, is our lexical database in this study. WordNet version 2.0 is utilized for extraction of the synsets of pattern couples in the texts.

Data mining Tool Support Vector Machine (SVM) is the data mining module for the main classification part. Recent studies have compared the performance of various classification algorithms including SVM with linear kernel, SVM with polynomial kernel of various degrees, SVM with RBF kernel with different variances, k-nearest neighbor algorithm and Naive Bayes [1]. In these experiments, SVM with linear kernel was consistently the best performer. These results confirm the results of the previous studies by Yang and Liu [16], Joachims [17] and Forman [18]. Thus, in this study we prefer SVM with linear kernel as the classification technique which is supposed to give best results standalone in recent comparable studies [18, 19]. For our experiments we used the SVMlight system, which is a rather efficient implementation by Joachims [19] and has been commonly used in previous studies. Classification of SVM is performed as one-versus-all for all dataset topics [18].

For the preprocessing of the datasets; each document is parsed, non-alphabetic characters and mark-up tags are discarded, case-folding is performed, and stop-words are eliminated. We utilize the list of 571 stopwords used in the Smart system [2]. For the term weighting approach, tf-idf technique is selected. The comparative study of different term weighting approaches in text retrieval have concluded that the commonly used tf-idf weighting outperforms other types [20]. Each document vector is normalized to account for documents of different lengths [1].

3.2 Dataset Selection

UCI Machine Learning Repository is inquired and standard Reuters - 21578 (Reuters) and National Science Foundation Research Award Abstracts (NSF) datasets are selected for our study [21]. The reason for this selection is that both datasets hold sentence information and the style of the texts are formal which has ordered sentences. These properties are crucial for efficient parsing in our IE approach for TC.

Reuters is a well-known dataset, which has been used for many TC algorithms [1, 16]. Standard ModApte split is used in which there are 9,603 training documents and 3,299 test documents. All the topics that exist both in the training and the test sets are utilized in the experiments. Our dataset thus consists of 90 classes and is highly skewed. For example, most of the classes have less than ten documents while seven classes have only one document in the training set. Also it allows multiple topics, which mean that documents in the corpus may belong to more than one existing topic.

NSF dataset consists of 129,000 abstracts describing NSF awards for basic research between the years 1990 and 2003 [21]. Due to this huge size, year 2001 is selected randomly and five sections (four sections for training and one section for test) are picked out from this year. Totally, there are 2368 training documents and 610 test documents with 122 existing topics. NSF is also skewed and allows multiple topics like Reuters.

3.3 Test Scenarios

Our main goal in this paper 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 AllWords Analysis and AllWordsPlus Analysis, respectively.

Five distinctive stemming styles are employed according to stemmer choice and WordNet utilization for both stages to be used in both AllWords Analysis and AllWordsPlus Analysis:

1. Raw Form: No stemming process is implemented for the classification algorithm and the words are used in their raw forms.
2. Only Porter Form: Morphological Porter stemmer is used for stemming.
3. Only Stanford Form: Syntactic Stanford stemmer is used for stemming.
4. WordNet Synsets Form: After stemming by the Stanford stemmer, WordNet is employed to extract the synset variations of all the words. Porter stemmer is not implemented as an alternative because output of this stemmer is not compatible for WordNet integration for two basic reasons. First, we need the correct POS information and the root form for our 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 (i.e. *earli* instead of *early*, *continu* instead of *continue*, etc.) required for the WordNet interface.
5. Stanford+Porter Form: Stanford stemmer is implemented initially for inflection removal, then Porter stemmer is used for the removal of derivational affixes of the same word.

AllWords Analysis In this stage, classical bow approach is implemented with the above stemming variations in order to find the optimal strategy for the bow approach. An example sentence is represented in Fig. 1.a. According to our classification in the previous paragraph, Fig. 1.b shows *Raw Form* for the example sentence without any stemming process. *Only Porter Form* and *Only Stanford Form* are represented in Fig. 1.c and Fig. 1.d. Fig. 1.e represents *WordNet Synsets Form* while *Stanford+Porter Form* is shown in Fig. 1.f.

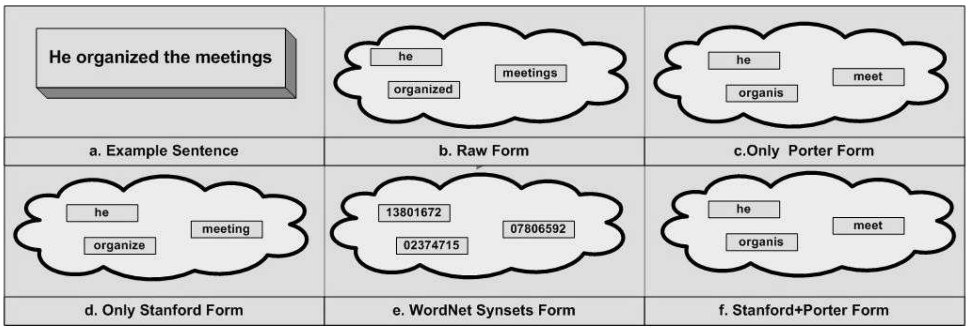

Fig. 1. Sample Keyword Formations due to Stemming Alternatives for AllWords Approach

AllWordsPlus Analysis In the second stage, we perform the re-examination of stemming alternatives; this time, not for the words but for the pattern couples. From another related perspective, we extend the bow approach by including the pattern couples which are varied by the stemming alternatives as shown in Fig. 2. For the required format of the bow approach for all words, we use the optimal solution which can be extracted from AllWords Analysis as seen in Fig. 2.a. According to our classification of stemming alternatives, Fig. 2.b shows *Raw Form* utilization for the patterns in addition to the optimally stemmed words for the example sentence. *Only Porter Form* and *Only Stanford Form* are used for pattern stemming as shown in Fig. 2.c and Fig. 2.d, respectively. Fig. 2.e represents *WordNet Synsets Form* while *Stanford+Porter Form* is shown in Fig. 2.f. Briefly, we use the optimally preprocessed *allwords* in the documents as the base keyword features for our algorithm and extend it with the pattern variations in each alternative implementation.

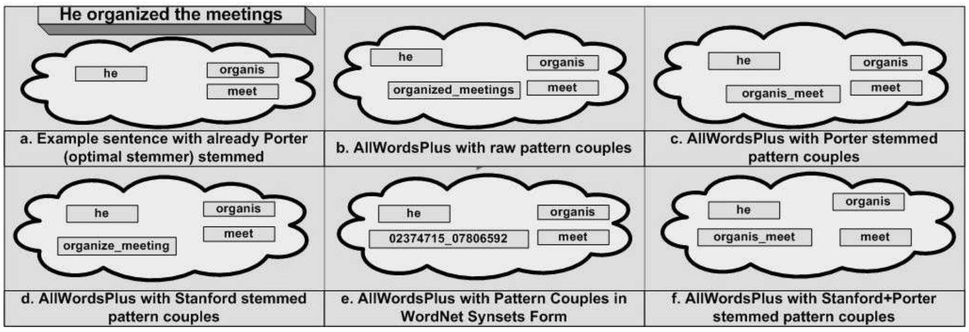
He organized the meetings
he
organis
meet
a. Example sentence with already Porter (optimal stemmer) stemmed
he
organized_meetings
organis
meet
b. AllWordsPlus with raw pattern couples
he
organis_meet
organis
meet
c. AllWordsPlus with Porter stemmed pattern couples
he
organize_meeting
organis
meet
d.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

Fig. 2. Sample Keyword Formations due to Stemming Alternatives for Patterns in AllWordsPlus Approach

4 Test Results

4.1 Evaluation Metrics

In order to evaluate the performance of our implementation we use the commonly used F-measure metric, which is equal to the harmonic mean of recall and precision [1]. F-measure score can be computed by two different alternatives, Micro-averaged F-Measure (MicroF) and Macro-averaged F-Measure (MacroF). MicroF gives equal weight to each document and is therefore considered as an average over all the document/category pairs while MacroF gives equal weight to each category so it is influenced more by the classifier's performance on rare categories. Keyword number (Key#) is another performance criterion which is the number of selected features for the solution vector of SVM.

4.2 Results and Discussion

Table 2. 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

AllWords Analysis The results for Allwords analysis is shown in Table 2. As can be seen in the table, morphological stemming of Porter Stemmer is found to be the optimal approach (*Raw* form outperforms it in Reuters but far behind it in NSF with also much more keywords) for all words stemming in both datasets, with low keyword numbers and high success rates. So, Porter stemmer is selected for stemming process of the words in our dataset for the next step.

Stanford stemmer, which is a more complicated syntactic parser, has mainly lower success rates with much more keyword numbers with respect to Porter stemmer. The main reason for this difference is that the outcomes of the stemmers are different in many cases. Stanford stemmer comes out with 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 stemmer finds the base forms as *arrival* and *arrive*, respectively. On the other hand, Porter stemmer finds the same base form *arrive* for both words by removing both the derivational affixes and inflections.

Table 3. Pattern Stemming

	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

AllWordsPlus Analysis For this approach, we can analyze the results from two points of view. First, we compare all the stemming approaches in our pattern utilized solution. We calculate the average of all pattern utilized results for each approach in both datasets. The comparison values are reported in Table 3. We see that, differences between stemming approaches is low, in terms of both keyword number and success rate, when compared with AllWords stemming alternatives summarized in Table 2. A possible reason for this low difference is the fact that pattern utilization integrates the related words in only certain forms to form couples, which decreases the role of stemming. In parallel to this idea, we can say that *raw* stemming process, which is the simplest form without any stemming process, is slightly better than the others in both datasets.

Table 4. Pattern Performance Ranks in Descending Order for Reuters and NSF

	Reuters	**Key#**	**MicroF**	**MacroF**		**NSF**	**Key#**	**MicroF**	**MacroF**
Rn	Patterns	avg	avg+/-std	avg±std	Rn	Patterns	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

In our second analysis, we compare the pattern utilization performance by analyzing the results through all stemming modes. Patterns are ranked (*Rn*) according to their MicroF and MacroF average scores (*avg*) with standard deviations (*std*) according to the alternative stemming modes in Table 4. As can be seen from the table, 14 patterns in Reuters and 15 patterns in NSF out of the possible 22 pattern types, have the power of outperforming the benchmark. The critical point is that, nine positive patterns achieve this performance in both datasets consistently with also very low standard deviations. Out of these nine *positive* patterns, we focus on the highest four patterns, namely : *part-participle modifier*, *adv-adverbal clause modifier*, *conj-conjunctive* and *poss-possession modifier*.

Fig. 3 shows the incremental effect of these patterns in Reuters and NSF. This improvement is shown by the dark colored part over the white color of the benchmark score for each pattern in the figure. These pattern types, when utilized standalone, improve the benchmark by around 0.4%-0.5%. *Part* pattern is the integration of a participle which is a derivative of a non-finite verb and the word modified by this participle (e.g. They compared it with *estimates derived* from ...). *Adv* pattern integrates the predicate with the supplementary word in the adverbial clause (e.g. We *must quickly* open our markets). *Conj* pattern relates the words which are connected by conjuncts (e.g. *Businessmen* and *officials* said that ...) while *poss* pattern relates the words by the possession feature (e.g. Japan has raised fears among many of *Asia's* exporting *nations...*). According to our observations, the common property of these patterns seem that they contain, not usually the word couples belonging to the closed classes (e.g. preposition, conjunction, etc.), but mostly the open class (e.g. noun, verb, etc.) words, which hold mutual characteristic relations within the couple. Another common feature is their frequencies which are adequate when compared with the number of all words in the datasets. For example, there are approximately 14,000 *conj*, 6,500 *adv*, 3,500 *part* and 1500 *poss* distinct patterns to be integrated with about 15,000 words in NSF. On the other hand, for example, *Attr* pattern with only 15 keywords performs ineffectively when integrated with all the words in this dataset.

There is also consistency for the three most unsuccessful pattern types in both Reuters and NSF: *preposition modifier - prep*, *noun compound modifier - nn*, and *auxiliaries-aux. Aux* pattern 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 pattern structure is generic (e.g. : make-is, running-are) which is not feasible for the classification problem. *Nn* pattern is the second worst pattern with low scores in both MicroF and MacroF measures. *Prep* pattern is an interesting pattern with leading scores in MicroF but lowest results with MacroF in both datasets.

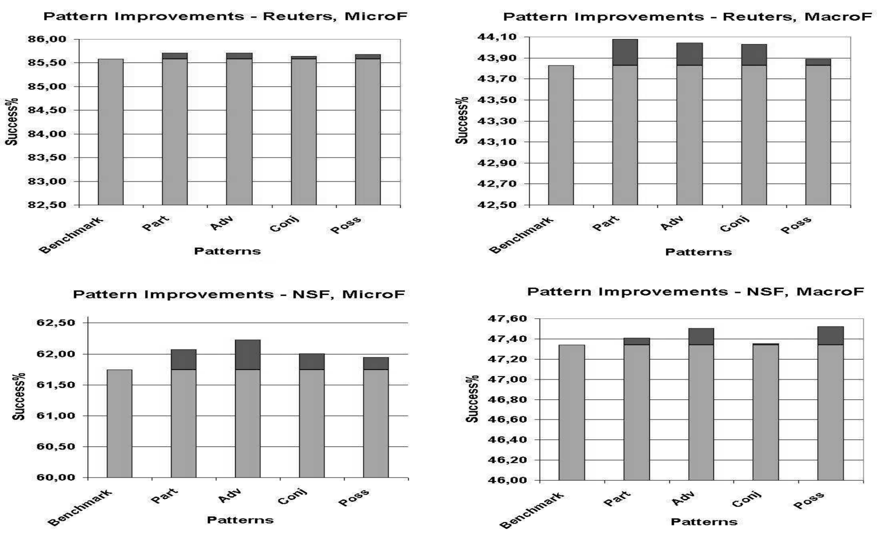

Fig. 3. Improvements of Successful Patterns Over the Benchmark

4.3 Hardware Specifications and Time Complexities

All experiments were implemented in Hp Workstation xw6200 with Xeon CPU 3.2 GHz and 4 GByte RAM.

Dataset parsing is the most time consuming part of the overall process which takes more than 10 hours for both datasets. However, this parsing operation is performed only once to be utilized for all the test modes for that dataset.

For a single test mode, the most time consuming part is the creation of tf-idf values for all existing terms in the training and test phases. This process takes approximately 90 minutes with about 30,000 keywords.

Due to memory and time limitations, maximum keyword number was limited with 36,000.

5 Conclusion and Future Work

To the best of our knowledge, this is the first study to fully explore all possible dependency patterns during the formation of the solution vector in the TC problem. The benchmark of the classical approach in TC is improved by the support of pattern utilization and further analysis can be performed to increase the improvement. In this performance, support of four patterns achieve the highest ranks, namely: *participle modifier*, *adverbal clause modifier*, *conjunctive* and *possession modifier* patterns. These pattern types improve the benchmark by around 0.4%-0.5%. There is also consistency for the three most unsuccessful pattern types in both Reuters and NSF. We have the motivation to analyze the syntactic and semantic features of both successful and unsuccessful patterns in more detail in order to utilize them more efficiently in our problem to increase the improvement.

Another contribution of this paper is its approach in stemmer utilization. Stemming is performed not only for the words but also for all the extracted pattern 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 pattern stemming.

We are planning to examine the use of selected keywords instead of all the words in the dataset for this problem. We consider to use the same distinction for keyword selection as the stemming approach in this study: keywords of all words and keywords of patterns. Selecting keywords separately from these groups and then combining them may yield better performance in terms of accuracy and time.

Acknowledgements. This work has been supported by the Boğaziçi University Research Fund under the grant number 05A103D.

References

1. Ozgur, A.: Supervised and Unsupervised Machine Learning Techniques for Text Document Categorization. MS Thesis. Bogazici University (2004)
2. Salton, G., Yang C.S., Wong A.: A Vector-Space Model for Automatic Indexing. Communications of the ACM 18 no.11 (1975)

3. Miller, G.: WordNet: a lexical database for English. Communications of the ACM, Volume 38, Issue 11 , pp.39-41 (1995)
4. Mansuy, T., Hilderman, R.: A Characterization of WordNet Features in Boolean Models for Text Classification. In: Proceedings of the 5th Australasian Data Mining Conference (AusDM'06), Sydney, Australia, November, pp. 103-109 (2006)
5. Moschitti, A., Basili, R.: Complex Linguistic Features for Text Classification. In: A Comprehensive Study. ECIR 2004. pp. 181-196 (2004)
6. Hidalgo, J.M.G, Rodriguez, M.B: Integrating a Lexical Database and a Training Collection for Text Categorization. In: ACL/EACL Workshop on Automatic Extraction and Building of Lexical Semantic Resources for Natural Language Applications (1997)
7. Bloehdorn, S., Hotho, A.: Boosting for text classification with semantic features. In: Proceedings of the MSW 2004 workshop at the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 70-87 (2004)
8. Bloehdorn, S., Moschitti, A.: Combined Syntactic and Semantic Kernels for Text Classification. In: ECIR 2007:307-318 (2007)
9. Porter, M.: An Algorithm for Suffix Stripping. In: Program 14, 130–137 (1980)
10. Stevenson, M., Greenwood, M.: A Semantic Approach to IE Pattern Induction. In: Proceedings of the 43rd Annual Meeting of the ACL, Ann Arbor (2005)
11. Stevenson, M., Greenwood, M.: Comparing Information Extraction Pattern Models. In: Proceedings of the Workshop on Information Extraction Beyond the Document, pp. 12-19, Sydney (2006)
12. Riloff, E.: Automatically Generating Extraction Patterns from Untagged Text. In: Proceedings of the 13th National Conference on AI - AAAI-96, pp. 1044-1049 (1996)
13. Marneffe, M.C., MacCartney, B., Manning, C.: Generating Typed Dependency Parses From Phrase Structure Parses. In: LREC2006 (2006)
14. Nastase, V., Shirabad, J.S.,Caropreso, M.F.: Using Dependency Relations for Text Classification. In: AI 2006, the nineteenth Canadian Conference on Artificial Intelligence, Québec City, Quebec, Canada (2006)
15. Klein, D., Manning, C.: Fast Exact Inference with a Factored Model for Natural Language Parsing. NIPS, volume 15. MIT Press (2003)
16. Yang, Y., Liu, X.: A Re-examination of Text Categorization Methods. Proceedings of SIGIR-99. In: 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley (1999)
17. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: European Conference on Machine Learning (ECML) (1998)
18. Forman, G.: An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research 3, 1289–1305 (2003)
19. Joachims, T.: Advances in Kernel Methods-Support Vector Learning. chapter Making Large-Scale SVM Learning Practical. MIT-Press (1999)
20. Salton, G., Buckley C: Term Weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24 no. 5, 513–523 (1988)
21. Asuncion, A., Newman, D.: UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science (2007)