Pattern Anal Applic (2012) 15:45–58
DOI 10.1007/s10044-010-0195-5

THEORETICAL ADVANCES

Optimization of dependency and pruning usage in text classification

Levent Özgür · Tunga Güngör

Received: 17 November 2009 / Accepted: 29 November 2010 / Published online: 23 December 2010

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

Keywords Text classification · Lexical dependency · Pruning analysis · Stanford parser

1 Originality and contribution

The motivation of this paper is to extend the standard bag-of-words (bow) method used in the text classification problem. The main contributions are twofold. On one hand, an analysis for the optimal level of pruning implementation of the features was performed by testing ten different levels. On the other hand, experiments with 36 lexical dependencies were performed independently and the final test was conducted using the combination of the leading dependencies in addition to all the words in the documents. Besides the benchmark bow method, three new methods concerning these subjects were proposed.

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, an extended version of the micro sign test was derived in this study. The results showed that for each extension in the methods, a corresponding significant improvement was observed in the success rates and pruning levels higher than the previously used standard level in the literature (i.e. two) were observed in almost all the experiments with the three datasets. In parallel with these results, the most advanced method combining the leading dependencies with optimal pruning levels outperformed all the other methods in terms of success rates with reasonable feature sizes.

As another contribution, the optimal feature numbers showed a consistent behavior (between 2,400 and 4,200) 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 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 in the leading dependency analysis, while the informal MiniNg20 benefited from different and simpler dependency types.

L. Özgür (✉) · T. Güngör
Department of Computer Engineering, Boğaziçi University,
Bebek 34342, Istanbul, Turkey
e-mail: ozgurlev@boun.edu.tr

T. Güngör
e-mail: gungort@boun.edu.tr

2 Introduction

Document patterns are domain specific and structured information which are extracted from unstructured

machine readable documents [1]. Different pattern types (predicate-argument model, chains, linked chains, subtrees, etc.) have been studied in the literature using alternative ways of linguistic analysis [2]. The aim in building a model based on document patterns is, in general, to extract sufficiently expressive patterns from the documents without causing too much additional complexity.

Lexical dependency is an extended model of document patterns in which sentence structure is represented using the grammatical relations (object-verb, conjunctive, prepositional modifier, etc.) between the words in a sentence [3]. A dependency is simply formed as the combination of any two words holding one 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 pattern 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 pattern, *dependency* is modified by the noun *classification* through the *in* preposition which forms the prepositional modifier dependency.

The concept of lexical dependency was previously used in many information retrieval applications such as syntactic tagging of words [4], parse disambiguation [5], text compression [6], machine translation [7], and textual entailment [8]. It was also employed as a framework for interactive and multilingual information retrieval problems that also include text classification implementation [9]. In this study, the aim is to perform a comprehensive analysis of dependency usage in the text classification (TC) domain. Basically, TC is a learning task in which documents are matched with category labels based on the likelihood suggested by a training set of labeled documents. Bag-of-words (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 solution vector used for classification. As an alternative to the standard bow approach, some semantic and syntactic-oriented solutions that utilize the WordNet system [10] were also proposed, but most of these solutions did not yield successful results due to the disambiguation problem [11]. There are also particular studies that aim to increase the success rate of the standard bow approach, but they either include semi-automatic processes [12] or significantly increase the complexity of the overall system [13] at the expense of more successful results.

There are some 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 the document representation; however, no significant improvement was achieved [14, 15]. 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 obtained; however, only the leading dependencies were used and the selection process benefited from human interaction (performed by human annotators) [16]. In another recent study, many linguistic features (e.g. part-of-speech information, word senses, proper nouns, etc.) were experimented in addition to the words, but no significant improvements were observed mostly due to the ambiguity problem [17]. Another related study reported the ineffectiveness of linguistic support in text classification by also pointing out the negative effect of a specific dependency: subject–object–verb [18]. There also exist studies based on dependencies that yielded successful results by using rule-based algorithms, but on a specific and not widely used dataset [19].

In almost all of these works, all dependency types were included in the solution vector together without any further analysis. Another drawback arises from the feature selection implementation which tries to filter unimportant and uninformative features based on some statistical ranking rules in order to reach more scalable and accurate solutions. Pruning, which is the simplest and one of the most efficient selection method, was mostly performed during the tests but with a predefined static level (e.g. two or three). A recent study increased the success rates of the classifier by considering noun-modifier dependencies and word senses in addition to the bow approach with also a fixed level of pruning implementation [20]. In another study which performed a distinct analysis of dependencies, a slight improvement over the baseline of the standard bow approach was achieved [21]. However, due to the lack of pruning, most of the dependency types used in this work 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. Besides pruning, there exist various types of feature selection metrics for filtering methods such as Chi-square, information gain, tf-idf, document frequency, etc. [22, 23]. Concerning these metrics, there are many studies analyzing and comparing the metrics [23, 24], combining them based on specific measurements [25], and providing unsupervised selection algorithms [26]. We chose pruning as the feature reduction approach in this work for its simplicity, efficiency, and success performance. In the initial tests, we also experimented with tf-idf as an alternative method. The success rate of the tf-idf approach when only the words were used as features was similar to that of the pruning implementation, but the results were not satisfactory when

the dependencies were included in the feature vector. Thus, we decided to continue with the pruning technique.

In this paper, we extend the dependency-oriented text classification studies with two main improvements: pruning analysis and combination of dependencies. The pruning implementation filters features having low frequencies in the datasets in order to arrive at fewer but more informative features. By this implementation, the aim is to solve the sparse solution vector problem caused by the excessive number of dependency features. Pruning is implemented incrementally in three main stages: pruning of words, pruning of dependencies, and pruning of dependency combinations. For the combination of dependencies, the leading dependencies that yield the most successful results for each dataset are combined and included in the solution vector as an extension to the standard bow approach.

The rest of the paper is organized as follows: Sect. 3 explains the system tools and environmental variables. The details of the proposed infrastructure are discussed in Sect. 4. The analysis of the results of the experiments are detailed in Sect. 5. The paper is concluded in Sect. 6.

3 System tools and environmental variables

In this section, the resources and the modules used in the proposed system are explained, which are the main building blocks of the system infrastructure. Such modules and resources are also used as standard tools in many other TC-related tasks.

3.1 Datasets

In this study, three datasets are used from the UCI machine learning repository: Reuters-21578 (Reuters), National Science Foundation research award abstracts (NSF), and mini 20 newsgroups (MiniNg20) [27]. Datasets with different characteristics were chosen in order to be able to analyze the effect of the methods on different types of data and to make comparisons between them.

Reuters is one of the most experimented datasets in TC studies [22, 28]. The standard Mod-Apte split is used which splits the dataset into 9,603 training documents and 3,299 test documents [22]. The dataset consists of 90 classes and is highly skewed. For instance, most of the classes have less than ten documents while seven classes have only one document in the training set. Also, the dataset allows multiple topics so that documents in the corpus may belong to more than one topic.

The NSF dataset consists of 129,000 abstracts describing NSF awards for basic research between the years 1990 and 2003 [27]. Five sections from the year 2001 were picked out randomly (four sections for training and one section for test). Five different groups were built, all the tests were repeated with these five cross folds, and their average was taken as the final result.

The MiniNg20 dataset consists of 2,000 messages with 1,600 training messages and 400 test messages that belong to a collection of 20 different usenet newsgroups. Unlike the other two datasets, this dataset is web-driven and informal with many misspellings, non-standard abbreviations, and many other text errors. It allows only one topic per message and is a balanced dataset having equal number of messages for each topic.

3.2 Preprocessing

Preprocessing is the initial step of text processing and it consists of standard routines such as removal of non-alphabetic characters and mark-up tags, case folding, elimination of stopwords, and stemming. Stoplists are used for automatic removal of uninformative words which causes a significant reduction in the number of features that have to be stored. In this work, the list of 571 stopwords of the Smart system was used for this purpose [22, 29]. Stemming was implemented using the Porter stemmer which is one of the most experienced stemmers for word forms [30]. Finally, term weighting was performed to assess the relative importance of the terms in the documents, which is fed to the machine learning component. As a common term weighting approach, the 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 Eq. 1.

$$tf - idf = tf_{t,d} \times \log \frac{N}{df_t}. \quad (1)$$

An alternative metric is Boolean weighting that checks only the occurrence of a term and does not consider the occurrence frequency. Thus, it is simpler than tf-idf and was outperformed by tf-idf in related studies [29, 31]. Another version of tf-idf, log(tf)-idf, was reported not to be significantly more successful than the original version [28], so the standard tf-idf metric was used in the proposed methods [22]. This standard formulation is used 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_t is the number of documents in the dataset that include t.

3.3 Machine learning tool

In text classification, we have a document space X, a description of each document $d \in X$, and a set of classes $C = c_1, c_2, \ldots, c_m$, where m is the number of classes.

Using machine learning algorithms, a classification function f is learned that maps documents to classes [22]:

$$f : X \rightarrow C. \quad (2)$$

The main machine learning approaches used in the TC domain may be classified as supervised vs. semi-supervised methods, parametric versus non-parametric methods, linear versus non-linear classifiers, vector space versus probabilistic classification, and decision tree modeling. 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.

Support vector machine (SVM) with linear kernel is the machine learning module which is used as the classification algorithm in this work. This is a supervised, linear and parametric vector space classification algorithm. Several studies have compared the performances of various classification algorithms including SVM with different kernels, k-nearest neighbor, and Naive Bayes in the text classification domain [23, 28, 32, 33]. Among these alternatives, SVM with linear kernel was shown to yield the best results. In the experiments, we use the $SVM^{\{light}$ system which is an efficient SVM implementation [33] that has been commonly used in previous studies. The one-versus-all mode is selected for dataset topics for SVM classification [23].

3.4 Syntactic tool

Stanford Parser is known to be one of the most powerful and efficient parsers having the least error rate [2]. 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 (PCFG), which was trained on the Penn Wall Street Journal treebank [34]. The part-of-speech tags of the tokens, the semantic heads in the sentence, and the dependents of the heads (auxiliaries, complement, etc.) are determined. Below the parse tree of the example sentence "We use combination of dependencies in text classification." is shown:

```
(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)))))))))
```

In the second phase, the dependencies extracted are labeled with grammatical relations using the tree-expression syntax defined by the tregex tool [35]. Below the dependencies obtained for the example sentence are listed (details of the dependencies will be explained in Sect. 4.2):

- Subject–verb (*We, use*)
- Object–verb (*combination, use*)
- Prepositional modifier-of (*combination, dependencies*)
- Noun compound modifier (*classification, text*)
- Prepositional modifier-in (*dependencies, classification*)

In the 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.

3.5 Evaluation criteria

3.5.1 F-Measure

In this work, to evaluate the performance of the proposed approaches, the commonly used F-measure metric is used, which is equal to the harmonic mean of precision (π) and recall (ρ) [22]. π and ρ are formulated as follows:

$$\pi_i = \frac{TP_i}{TP_i + FP_i}, \quad \rho_i = \frac{TP_i}{TP_i + FN_i} \quad (3)$$

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 [22].

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

$$F(\text{micro-averaged}) = \frac{2 \times \pi \times \rho}{\pi + \rho} \quad (4)$$

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:

$$F_i = \frac{2 \times \pi_i \times \rho_i}{\pi_i + \rho_i}, \quad F(\text{macro-averaged}) = \frac{\sum_{i=1}^{M} F_i}{M}, \quad (5)$$

where M is the 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. Both measurement scores are provided in the experiments to be more informative.

3.5.2 Significance tests

A set of three significance tests was designed to compare the proposed methods and to give an extensive and robust analysis of the results:

- *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 [28]. Standard z values are calculated and the corresponding confidence levels are determined according to the standard normal distribution [36, 37].
- *Micro sign test with positive instances*. 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, an extension of the micro sign test was derived 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.
- *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 [28]. The z value calculation and confidence level determination processes are the same as in the micro sign test.

4 Main infrastructure and experimental design

In this section, the details of the proposed approach are discussed. The main contributions in this research are the pruning implementation and the dependency usage, and they will be explained in Sects. 4.1 and 4.2, respectively. Alternative scenarios based on these concepts will be analyzed in Sect. 4.3.

4.1 Pruning implementation

Pruning is used 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). The pruning levels are analyzed for words, dependencies, and dependency combinations separately. $\mathrm{PL} = n(n \geq 1)$ 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 $\mathrm{PL} = 1$ means that no pruning is implemented for that feature type.

The pruning concept is especially useful for dependency features. As mentioned in Sect. 2, a dependency is formed by combining any two dependent words. Most of the dependency types (see Table 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 [21]. 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 parameter tuning is performed 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 Sect. 4.3.

4.2 Dependency analysis

36 dependency types are used in the tests. Table 1 shows the complete list of dependency types accompanied with their definitions and some examples. Dependency types formed of numeric tokens (i.e. *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

Table 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

the classifier. For instance, the types *dobj* (direct object), *iobj* (indirect object), and *pobj* (prepositional object) 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*).

4.3 Experimental design

An incremental framework is designed for the analysis of the dependency and pruning concepts in the TC domain. As can be seen in Fig. 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, the experiments are repeated with incremental PL values. We stop incrementing the PL value when 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.

The effect of pruning is observed to diminish 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 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 the PL value is incremented, for instance, from PL = 20 to 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, the pruning analysis is performed 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).

4.3.1 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. The 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.

4.3.2 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 and usually the pruning level is arbitrarily fixed to a small value such as two [20]. In this work, this method is implemented with several pruning levels (2, 3, 5, 8, 13, 20, and 30) to determine the optimal word PL value for each dataset.

4.3.3 AWDP

The AWDP (all words and dependencies with pruning) method extends both the AW and the AWP approaches 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. The PL values (2, 3, 5, 8, 13, 20, and 30) are used in this stage.

4.3.4 AWDCP

The AWDCP (all words and dependency combinations with pruning) method extends the AWDP method 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. Pruning level analysis is performed using ten 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.

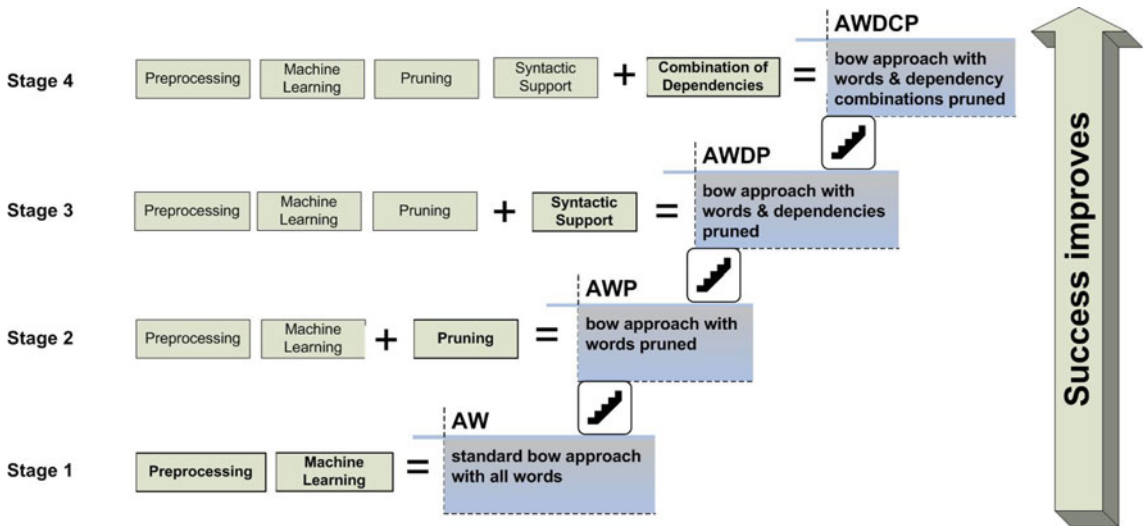

Fig. 1 General system architecture with the proposed methods

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 Analysis of results

In this section, first the optimal values of the pruning level parameter used in the methods are explained. Then the results of the experiments with these parameter values are discussed. 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.

5.1 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 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, most of the successful dependency types are observed to converge to similar optimal pruning levels.

For the AWDCP method, the five leading dependency types determined by AWDP for each dataset (see Table 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 Sect. 5.3.

5.2 Performance of the methods

Table 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 is used 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 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 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.

5.3 Pruning level analysis

Figures 2 and 3 show, respectively, 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 Sect. 5.2.

The curves in the figures are observed to follow a similar pattern with respect to the PL improvement. Although the optimal pruning level varies depending on the 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. Despite the previous related studies fixing the pruning level to a small value (e.g. two) [20], the results reveal that the optimal value is generally much higher than that value with almost all the proposed methods in the three datasets. A detailed analysis about the pruning levels with respect to the dataset properties will be given in Sect. 5.6.2.

5.4 Optimal feature numbers

Table 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 2,400 and 4,200.

In different studies related to feature selection in the literature, several feature number levels (500, 1,000, 2,000,

Table 2 Success scores of the proposed methods

	Reuters			NSF			MiniNg20		
	Key#	MicroF	MacroF	Key#	MicroF	MacroF	Key#	MicroF	MacroF
AWDCP	4,138	86.03	45.26	3,908	66.01	47.68	2,914	54.23	51.65
AWDP	4,198	85.96	45.07	2,829	65.07	47.10	3,114	54.13	51.53
AWP	3,976	85.84	44.85	2,478	64.58	46.49	2,863	53.62	51.02
AW	20,292	85.58	43.83	13,424	64.46	46.11	30,970	46.42	43.44

Table 3 Success scores of the leading dependencies in AWDP with the optimal PL values

	Reuters	PL Word: 13 PL Dep.: 8		NSF	PL Word: 13 PL Dep.: 8		MiniNg20	PL Word: 13 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

5,000, 10,000, etc.) were reported to give successful results with different machine learning algorithms [28, 29, 33]. The optimal feature number range that was obtained in this work (2,400–4,200) can be said to be consistent with these stated results.

5.5 Significance of the improvements

Table 4 compares the methods used in this work and shows the statistical improvement results. Three significance tests that have been defined in Sect. 3.5.2 were applied. The micro sign test measures the improvement over the whole document-category matrix. An extended version of this measure (micro sign test with positive instances) has been derived 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 looked up in the z-table. The following symbols and terminology are used 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%)

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. Each method is observed to significantly outperform its predecessor method (AWDCP $\gg$ AWDP $\gg$ AWP $\gg$ AW) and AWDCP (the most advanced method proposed in the study) is significantly the best method.

5.6 Dataset comparison

As discussed in Sect. 3.1, all the experiments were performed with three different datasets. 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, the results are analyzed from the dataset perspective and the datasets are compared according to these characteristics and results.

5.6.1 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 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 Sect. 3.5.1, 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

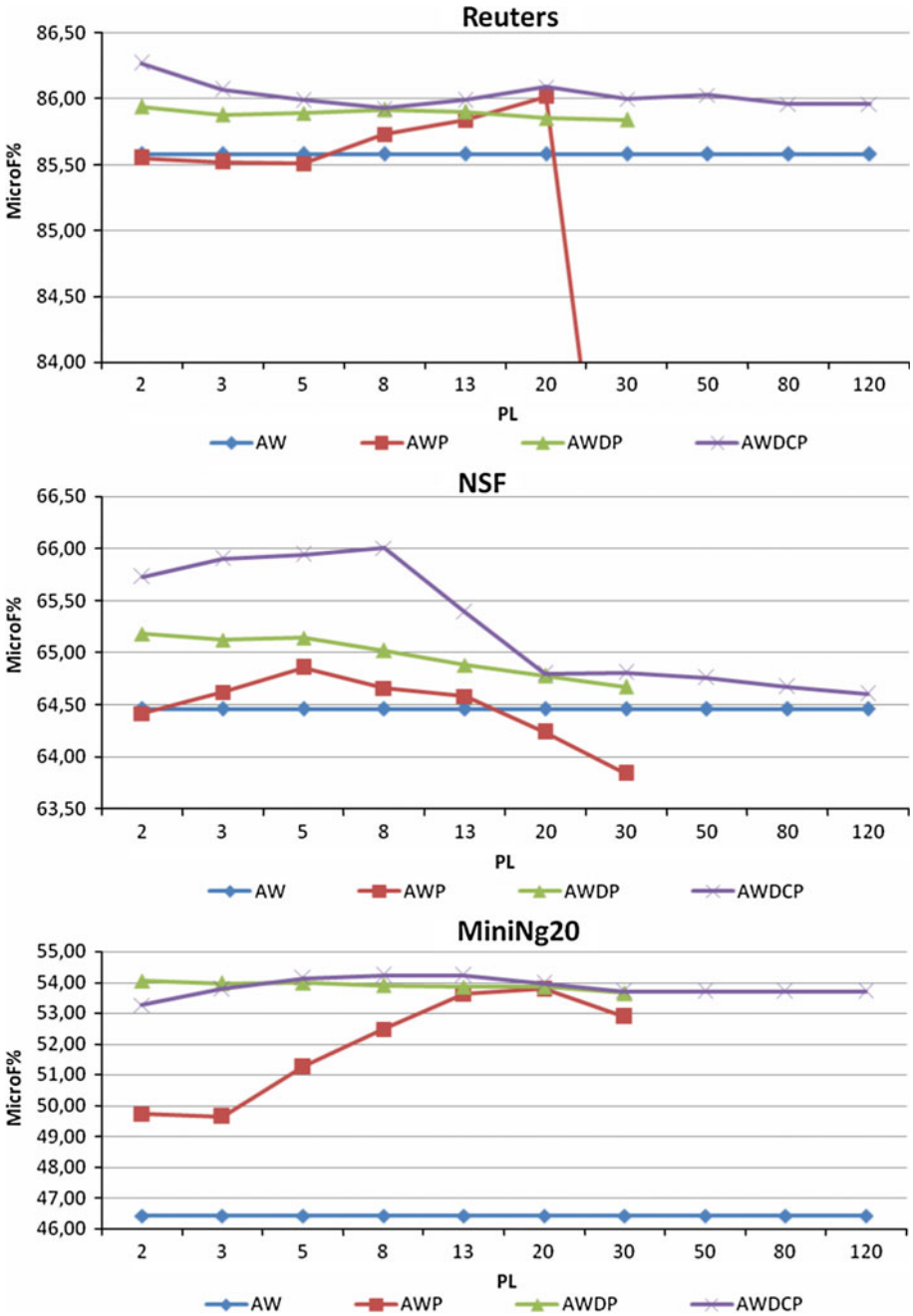

Fig. 2 Pruning level analysis with the proposed methods (MicroF)

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 does not exist available 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.

5.6.2 *Optimal PL values*

The optimal PL value shows variation as a function of the applied method and the dataset. When the pruning levels are compared with respect to the methods, an expected pattern is observed. 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

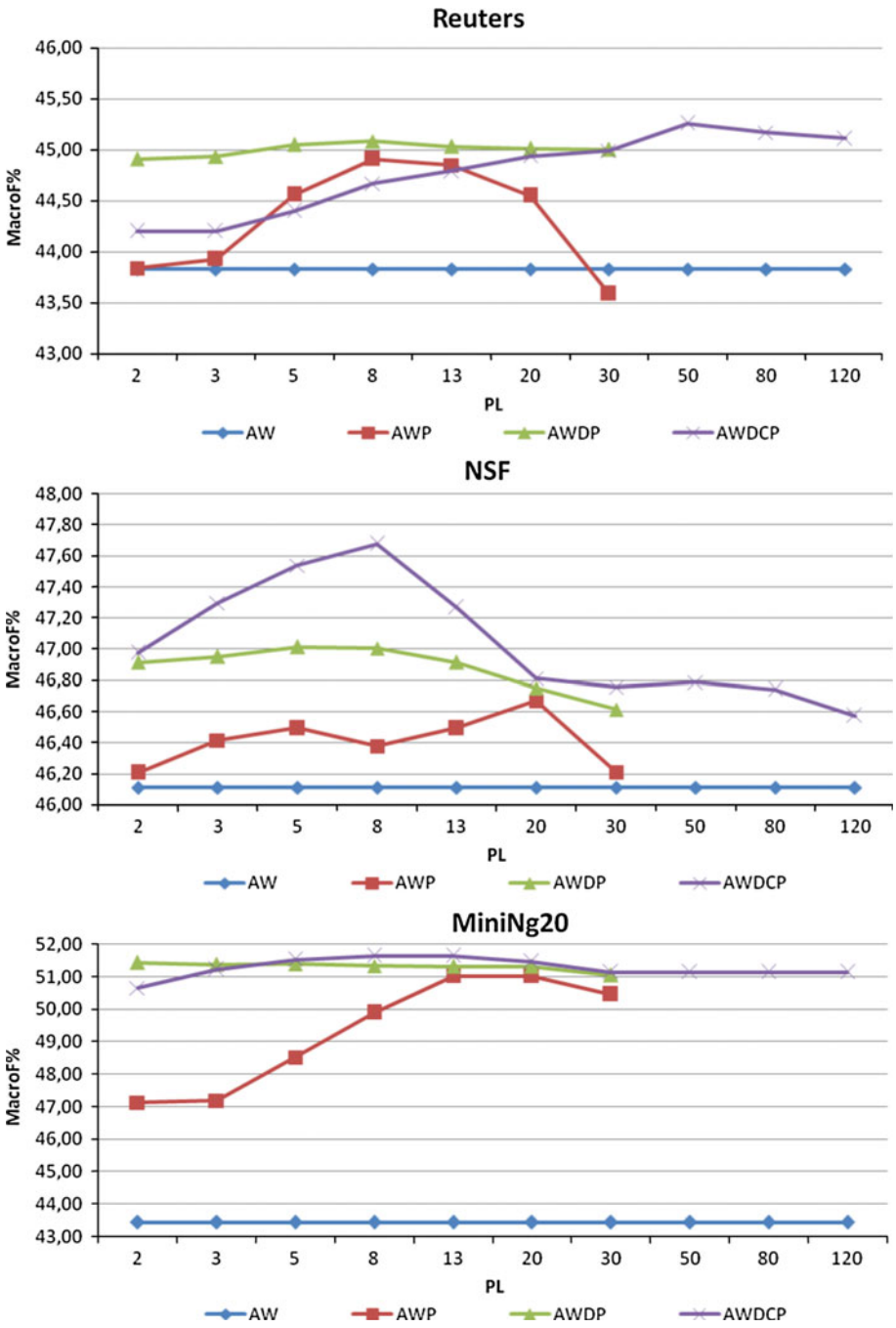

Fig. 3 Pruning level analysis with the proposed methods (MacroF)

AWDCP methods. AWDCP includes in the feature vector 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 the optimal values were fixed 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 the AWDCP method is applied, 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 in fact understandable because in the AWDP tests with NSF, PL value may have been selected as 3 or 5

Table 4 Statistical comparison of the proposed methods

	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	$\gg$	$\gg$	$>$

(which, in the next stage, would yield the improvement of AWDCP) but the larger value was preferred (PL = 8, which gave 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 more 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.

5.6.3 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 affected by the grammatical level of a document, less accurate parse results are achieved 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 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.

5.6.4 Common successful dependencies

Table 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 this dependency, *prt* (phrasal verb participle, e.g. *write down*) which is one of the simplest dependencies yields the most successful results with MiniNg20. Although much more tests with different dataset types are needed to perform automatic detection of the most useful dependencies, it seems that dependency complexity is positively correlated with the dataset formality level: the more formal a dataset is, the more complicated dependencies it benefits from.

6 Conclusion

The main motivation of this paper 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, the concepts of lexical dependencies and pruning were incorporated into the algorithms and the optimal parameter values were determined 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. SVM was used for the machine learning component, which is a state-of-the-art classifier in TC, and the Stanford Parser was used as the syntactic tool. All the experiments were repeated in three different datasets (Reuters, NSF, and MiniNg20).

Three significance tests have been implemented including the extended version of the micro sign test that

has been 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 with reasonable feature sizes. According to the results, that optimal pruning level was generally found to be much more than PL = 2 (the standard fixed value for pruning in related studies) with almost all the proposed methods in the three datasets. The optimal feature numbers have been observed to show a consistent behavior (between 2,400 and 4,200) 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 and simpler dependency types as the leading ones.

As future work, one possible direction is incorporating feature selection algorithms into the proposed methods. In this study, pruning implementation was implemented for feature filtering but feature selection is different from this filtering process using specific methods such as information gain, tf-idf, etc. These algorithms will be implemented in accordance with the pruning implementation and dependency usage for text classification in the future studies. Another possible related study is to apply the same tests to more datasets with different skewness and formality levels in order to develop robust algorithms for automatic detection of optimal pruning levels and most useful dependencies according to dataset properties.

Acknowledgments This work was supported by the Boğaziçi University Research Fund under the Grant Number 05A103D and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.

References

1. Stevenson M, Greenwood M (2005) A semantic approach to IE pattern induction. In: Proceedings of the 43rd annual meeting of the ACL, Ann Arbor
2. Stevenson M, Greenwood M (2006) Comparing information extraction pattern models. In: Proceedings of the workshop on information extraction beyond the document, Sydney, pp 12–19
3. Marneffe MC, MacCartney B, Manning C (2006) Generating typed dependency parses from phrase structure parses. LREC2006
4. Finch A, Black A, Hwang YS, Sumita E (2006) Using lexical dependency and ontological knowledge to improve a detailed syntactic and semantic tagger of English. In: Proceedings of the COLING/ACL on main conference poster sessions, Sydney, pp 215–222
5. Cahill A, Heid U, Rohrer C, Weller M (2009) Using tri-lexical dependencies in LFG parse disambiguation. In: The 14th international LFG conference, July 2009. Trinity College, Cambridge
6. Bach J, Witten IH (1999) Lexical attraction for text compression. In: Proceedings of the conference on data compression, DCC 1999
7. Charniak E et al. (2003) Syntax-based language models for statistical machine translation. In: Proceedings of the MT summit 2003
8. Herrera J, Penas A, Verdejo F (2006) Textual entailment recognition based on dependency analysis and WordNet. In: Lecture notes in computer science, vol 3944/2006. Springer, Berlin
9. Basili R, Pazienza MT, Mazzucchelli L (2000) An adaptive and distributed framework for advanced IR. RIAO 2000, pp 908–922
10. Miller G (1995) WordNet: a lexical database for English. communications of the ACM, vol 38, no 11, pp 39–41
11. Mansuy T, Hilderman R (2006) A characterization of WordNet features in Boolean models for text classification. In: Proceedings of the 5th Australasian data mining conference (AusDM'06), Sydney, November, pp 103–109
12. Hidalgo JMG, Rodriguez MB (1997) 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
13. Bloehdorn S, Moschitti A (2007) Combined syntactic and semantic kernels for text classification. ECIR 2007, pp 307–318
14. Lewis DD (1992) An evaluation of phrasal and clustered representations on a text categorization task. In: Proceedings of SIGIR-92, Copenhagen, pp 37–50
15. Furnkranz J, Mitchell T, Rilof E (1998) A case study in using linguistic phrases for text categorization on the WWW. AAAI-98 workshop on learning for text categorization
16. KÖnig AC, Brill E (2006) Reducing the human overhead in text categorization. In: Proceedings of KDD 2006, Association for Computing Machinery Inc
17. Moschitti A, Basili R (2004) Complex linguistic features for text classification. In: A comprehensive study. ECIR 2004, pp 181–196
18. Moschitti A (2008) Kernel methods, syntax and semantics for relational text categorization. In: Proceeding of ACM 17th conference on information and knowledge management (CIKM), Napa Valley
19. Ghanem M, Guo Y, Lodhi H, Zhang Y (2002) Automatic scientific text classification using local patterns. In: KDD CUP 2002 (Task1), SIGKDD Explorations, vol 4, no 2, pp 95–96
20. Nastase V, Shirabad JS, Caropreso MF (2006) Using dependency relations for text classification. In: AI 2006, the nineteenth Canadian conference on artificial intelligence, Quebec City
21. Özgür L, Güngör T (2009) Analysis of stemming alternatives and ddependency pattern support in text classification. In: CICLing 2009, the tenth international conference on intelligent text processing and computational linguistics. Research in computing science, vol 41, Mexico City, Mexico.
22. Manning C.D, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
23. Forman G (2003) An extensive empirical study of feature selection metrics for text classification. J Mach Learn Res 3:1289–1305

24. Yang Y, Pedersen JO (1997) A comparative study on feature selection in text categorization. In: Proceedings of the 14th international conference on machine learning, pp 412–420
25. Shoushan L, Rui X, Chengqing Z, Huang CR (2009) A framework of feature selection methods for text categorization. In: Proceedings of the 47th annual meeting of the ACL and the 4t IJCNLP of the AFNLP, Suntec, pp 692–700
26. Dasgupta A, Drineas P, Harb B, Josifovski V, Mahoney MW (2007) Feature selection methods for text classification. In: Proceedings of 13th annual SIGKDD, pp 230–239
27. Asuncion A, Newman D (2007) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine. http://www.ics.uci.edu/mlearn/MLRepository.html
28. Yang Y, Liu X (1999) A Re-examination of text categorization methods. In: Proceedings of SIGIR-99. 22nd ACM international conference on research and development in information retrieval, Berkeley
29. Özgür A, Özgür L and Güngör T (2005) Text categorization with class-based and corpus-based keyword selection. Lecture notes in computer science, vol 3733. Springer, Berlin, pp 606–615
30. Porter M (1980) An algorithm for suffix stripping. In: Program 14, pp 130–137
31. Salton G, Buckley C (1988) Term weighting approaches in automatic text retrieval. Inf Process Manage 24(5):513–523
32. Joachims T (1999) Advances in kernel methods-support vector learning. Making large-scale SVM learning practical. MIT Press, Cambridge
33. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: European conference on machine learning (ECML). Springer, Berlin, pp 137–142
34. Klein D, Manning C (2003) Fast exact inference with a factored model for natural language parsing, vol 15, NIPS. MIT Press, Cambridge
35. Levy R, Andrew G (2006) Tregex and Tsurgeon: tools for querying and manipulating tree data structures. 5th international conference on language resources and evaluation (LREC 2006)
36. Larson R, Farber B (2000) Elementary statistics: picturing the World. Prentice Hall, Englewood Cliffs
37. Montgomery DC (2001) Design and analysis of experiments. Wiley, New York