Pattern Recognition Letters xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Text classification with the support of pruned dependency patterns

Levent Özgür, Tunga Güngör *

Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

ARTICLE INFO

Article history:
Received 5 June 2009
Available online xxxx

Communicated by C.L. Tan

Keywords:
Text classification
Lexical dependency
Pruning analysis
Reuters-21578

ABSTRACT

We propose a novel text classification approach based on two main concepts, lexical dependency and pruning. We extend the standard bag-of-words method by including dependency patterns in the feature vector. We perform experiments with 37 lexical dependencies and the effect of each dependency type is analyzed separately in order to identify the most discriminative dependencies. We analyze the effect of pruning (filtering features with low frequencies) for both word features and dependency features. Parameter tuning is performed with eight different pruning levels to determine the optimal levels. The experiments were repeated on three datasets with different characteristics. We observed a significant improvement on the success rates as well as a reduction on the dimensionality of the feature vector. We argue that, in contrast to the works in the literature, a much higher pruning level should be used in text classification. By analyzing the results from the dataset perspective, we also show that datasets in similar formality levels have similar leading dependencies and show close behavior with varying pruning levels.

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 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. It ignores the order of the words and the relations between the words and simplifies the architecture by directly focusing on only the frequency of the terms.

In this study, we extend the bow approach for text classification with two main concepts: *lexical dependency* and *pruning*. Lexical dependency is a kind of document pattern that shows explicitly the grammatical relations (object, preposition, etc.) within a sentence. In the dependency analysis stage, we use 37 different types of lexical dependencies to enrich the bow-oriented feature vector which is normally composed of only words. The effect of each dependency is analyzed separately in order to extract the most discriminative dependency types for the TC problem. In this way, we aim at improving the standard approach by taking the relations between the words into account.

The pruning process basically filters less frequent features in a document collection in order to arrive at fewer but more informative features. In this work, we perform a comprehensive pruning analysis and parameter tuning to find the optimal level of pruning. We employ two types of features (word features and dependency features) and we obtain different pruning levels for each. For all the datasets used in this research, the optimal pruning level for words was found as 13 among the tested values. Pruning was performed also for lexical dependencies and the optimal pruning levels were determined as 8 for the Reuters and NSF datasets and 2 for the MiniNg20 dataset.

In this work, we use the standard bow approach as the baseline method. In addition to the bow approach, we propose two models: a model that incorporates the word pruning concept into the baseline model, and a model that makes use of lexical dependencies in addition to words and that performs pruning for both words and dependencies. We show that the proposed two models are significantly more successful than the standard bow approach. Also, the last model which is based on both the dependency and the pruning concepts shows a statistical improvement over the model that uses word pruning only.

The rest of the paper is organized as follows: Section 2 gives a summary of related work. We discuss the details of the proposed system in Section 3. The experiment results and the implication of these results are detailed in Section 4. We conclude the paper in Section 5.

2. Related work

We can classify the related work in two main groups: studies in text classification and studies on the dependency concept.

* Corresponding author. Tel.: +90 212 3597774; fax: +90 212 2872461.
E-mail addresses: ozgurlev@boun.edu.tr (L. Özgür), gungort@boun.edu.tr (T. Güngör).

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.05.005

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

2.1. Text classification approaches

Most of the studies aimed at solving the TC problem implement the bow structure. Using a machine learning algorithm that considers the terms in the training and test data as the basic features is the fundamental and conventional architecture for the text classification problem (Manning et al., 2008; Yang and Liu, 1999). In this approach, documents are represented by the widely-used vector-space model introduced by Salton et al. (1975). 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 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 were proposed to compensate for the ordering information and yielded promising results (Lodhi et al., 2002). But this method has to deal with performance problems in large datasets – it suffers big space and time complexities and thus uses 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 accepted as too complicated models for text classification (Manning et al., 2008). These models are more appropriate for problems like query generation from texts, speech recognition, etc.

Main machine learning approaches used in the TC domain may be classified as supervised (e.g. support vector machine) vs. semi-supervised (e.g. using naive Bayes with expectation maximization) methods, parametric (e.g. support vector machine, naive Bayes) vs. non-parametric (e.g. k-nearest neighbor) methods, linear (e.g. support vector machine with linear kernel) vs. non-linear (e.g. support vector machine with radial basis kernel) classifiers, vector space (e.g. 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. Several studies have compared the performances of these approaches and in general support vector machine (SVM) with linear kernel was shown to yield the leading results (Yang and Liu, 1999; Joachims, 1999; Forman, 2003; Özgür et al., 2005). 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 additive 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 (Joachims, 1999).

2.2. Dependency concept

Information extraction (IE) discipline aims at extracting structured information from unstructured machine-readable documents. A critical problem in IE is to develop systems which can be easily adapted to new domains as automatically and correctly as possible (Stevenson and Greenwood, 2005). Solutions to this problem attempt to learn domain-specific information, represented in the form of document patterns. Patterns can be structured in many different ways with different levels of linguistic analysis. In a detailed study on pattern structures, four different pattern models were analyzed which are predicate-argument model (SVO), chains, linked chains, and subtrees (Stevenson and Greenwood, 2006).

Table 1
Dependencies in the sentence "*We use dependencies in text classification*".

Dependency type	Word pair
subject-verb	we-use
object-verb	dependencies-use
noun compound modifier	classification-text
prepositional modifier	dependencies-classification

Lexical dependency is an extended model of SVO patterns: sentence structure is represented using grammatical relations between the words in a sentence (object-verb, conjunctive, prepositional modifier, etc.) (Marneffe et al., 2006). A dependency is simply formed by the combination of any two words holding one of these grammatical relations. Table 1 shows the lexical dependencies extracted from an example sentence.

The concept of lexical dependency was previously used in many information retrieval applications such as sentiment analysis (Mullen and Collier, 2004), parse disambiguation (Cahill et al., 2009), machine translation (Charniak et al., 2003), textual entailment (Herrera et al., 2006), and discourse coherence (Wellner et al., 2006). It was also employed as a common framework for interactive, multimodal, and multilingual information retrieval problems that also included text classification implementation (Basili et al., 2000).

There are a number of studies that specifically focus on the use of dependencies in text classification. Pioneering studies in this topic included noun phrases and main argument dependencies (subject-verb, object-verb, etc.) in the classification algorithms, but no significant improvement was achieved (Lewis, 1992; Furnkranz et al., 1998). In a 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, but only the leading dependencies were used and the selection process required human interaction (König and Brill, 2006). In another study, some linguistic features (e.g. part-of-speech information, complex nominals, proper nouns, and word senses) were considered in addition to the words, but no significant improvement was observed (Moschitti and Basili, 2004). Later, by referring to the negative effect of a specific dependency (subject-object-verb), Moschitti (2008) mentioned that linguistic processing does not improve the bow approach in general. A related study extracted dependencies by capturing frequently occurring keyword combinations within short segments using a rule-based algorithm (Ghanem et al., 2002). The algorithm yielded successful results but the experiments were done only on a specific and not widely used dataset. Another study increased the success rates of the classifier by accompanying the bow approach with a combination of noun-modifier dependencies and word senses (Nastase et al., 2006).

In almost all of these studies, dependencies were included in the solution vector together without a further and specific analysis of each dependency type. Another drawback was about pruning. Most of these works used pruning during the tests, but the pruning threshold was set to a pre-defined level without an analysis of the optimal level. In a recent study which performed a distinct analysis of dependencies, a slight improvement over the baseline of the standard bow approach was achieved (Özgür and Güngör, 2009). However, due to the lack of pruning implementation, most of the dependency types used yielded many instances (distinct word pairs), which caused an excessive number of features and a highly sparse solution set in the machine learning algorithm.

3. Methodology

The main aim in this study is to extend the standard bow approach for the text classification problem by making use of the

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

lexical dependency concept with varying levels of pruning. In order to reach robust results about the impact of the proposed approaches and generalize the outcomes, we use three independent datasets in the experiments and perform significance tests. In this section, we cover the details of the proposed solution.

3.1. Core idea

We basically implement three main approaches in this study. Fig. 1 shows the general system architecture corresponding to these approaches. AW (all words) is the baseline method that uses the standard bow approach with all the words in the feature vector. In the other two methods, we perform pruning of the features and filter those having frequencies below a threshold value. 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 (e.g. Nastase et al., 2006). 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 by pruning both of these feature types for the final feature set.

We use pruning in order to reach a smaller but more discriminative feature set to be used by the machine learning algorithm. For this purpose, we filter the terms that occur less than a certain threshold value in the whole training set. We name this threshold value as the *pruning level* (PL). PL = n ($n \geqslant 1$) indicates that features occurring at least n times in the training set are used in the solution vector while the others are ignored. Note that PL = 1 corresponds to the AW method (i.e. no pruning).

Our main motivation in this study is to extract the most successful features and use them in an optimal manner for the TC problem. This can be done either by filtering the features (pruning) with respect to a threshold value or by employing a feature selection metric (mutual information, chi square, tf-idf, etc.). Latent semantic indexing (LSI) is a feature extraction approach that also reduces the size of the feature vector, but differs from the selection methods by transferring the feature vector into a reduced representative set of features. The method represents the documents and terms in the same space by allowing the underlying semantic relationships between them (Wang and Zhang, 2006). 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 the classification performance (Liu et al., 2004).

We chose pruning as the feature reduction approach in this work which is the simplest and the most efficient method for this purpose. 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 (Manning et al., 2008). 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 as future work.

One of the main contributions of this study is that we perform parameter tuning by analyzing different values for each dataset to reach the optimal PL values for the AWP and AWDP methods. We conduct experiments with different pruning levels between 1 and 30: 1, 2, 3, 5, 8, 13, 20, and 30. Pruning for words and dependencies were analyzed separately since the optimal pruning levels would be different in each case. Since dependencies are formed as pairs of words, they occur with much less frequencies than words and thus they should be subjected to smaller PL values.

Table 2 shows the effect of the pruning process on a sample from the NSF dataset. The sample consists of features (words and dependencies) between *center* and *chao*. To simplify the example, only the *noun compound modifier* (*nn*) dependency is included

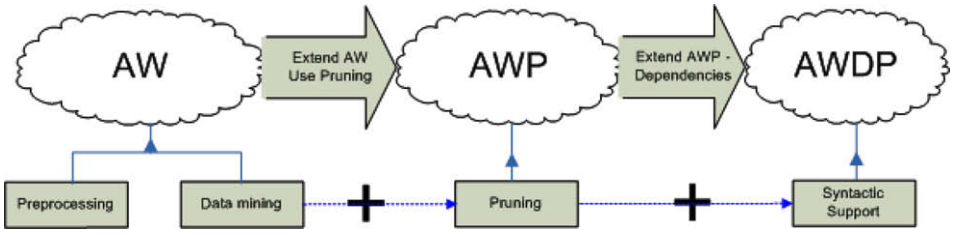

Fig. 1. General system architecture.

Table 2
Feature numbers and samples in NSF dataset (word PL: 13).

PL	F#	Sample feature set
1	229	center, center-abstract, center-accelerator, center-advanced, center-alto, center-analysis, center-arabidopsis, center-asilomar, center-aspen, center-berkeley, center-biological, center-biotechnology, center-bir,...,channels-calcium, channels-cdma, channels-communication, channels-dispersive, channels-ion, channels-potassium, channels-radio, channels-spin, channels-time, chan-professor, chan-sunney
2	62	center, center-aspen, center-berkeley, center-conference, center-cooperative, center-engineering, center-fes, center-geochronology, center-industry, center-international, center-limnological, center-materials,...,changes-climate, changes-ecosystem, changes-effect, changes-eocene, changes-level, changes-phase, changes-shape, changes-term
3	34	center, center-berkeley, center-cooperative, center-engineering, center-geochronology, center-industry,...,challeng, chamber, chang, change-climate, change-culture, change-global, change-scale, changes-calcium, changes-chemical, changes-climate, changes-phase, changes-term
5	20	center, center-engineering, center-national, center-research, centers-research, central, centrin, centuri, century-half, century-quarter, ceram, cerevisia, cerevisiae-yeast, cf, chain, challeng, chamber, chang, change-climate, changes-climate
8	18	center, center-engineering, center-national, center-research, centers-research, central, centrin, centuri, century-half, century-quarter, ceram, cerevisia, cf, chain, challeng, chamber, chang, change-climate
13	15	center, center-engineering, centers-research, central, centrin, centuri, century-quarter, ceram, cerevisia, cf, chain, challeng, chamber, chang, change-climate
20	13	center, centers-research, central, centrin, centuri, ceram, cerevisia, cf, chain, challeng, chamber, chang, change-climate
30	12	center, central, centrin, centuri, ceram, cerevisia, cf, chain, challeng, chamber, chang, change-climate

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

and PL for words is fixed as 13 (i.e. words that appear less than 13 times in the dataset are eliminated). Each row in the table corresponds to a PL value for dependencies and shows the number of features (F#) with this PL and the list of these features. When PL = 1, the feature set includes a large number of features (a total of 229 features among which 218 are lexical dependencies). Increasing the PL value by just one eliminates about 77% of the dependencies, indicating that most of the word pairs occur only once in the whole dataset. When PL = 30, only one dependency and 11 words remain in the feature set. We varied the pruning level with small increments for low PL values (e.g. PL = 1, 2, 3) and larger increments for high PL values (e.g. PL = 20, 30), since an increase in PL at high levels contributes less to the pruning process. This example shows the effect of the pruning process in detail in decreasing the size of the feature set.

3.2. Dependency types

A recent study about dependency support in text classification has analyzed 22 grammatical relations (Özgür and Güngör, 2009). In order to make a comparison with this study, we also use these relations in our proposed system. Besides these dependencies, we performed a linguistic analysis of dependencies and enriched the dependency usage with a set of linguistically-motivated decisions.

The lexical dependency *prepositional modifier* (*prep*) has the largest feature number, but it is one of the most unsuccessful dependencies. In general, prepositions provide an important function in sentences by integrating related words. However, this integration covers different contexts, hence there are many subtypes of this characteristic dependency. Regarding all these subtypes as the same and representing all with a single feature type causes confusion during classification. Based on this observation, we split this dependency into 15 possible subdependencies, each preserving its particular usage pattern, in order to understand whether this type of information has discriminative power in classification.

In the initial tests, we also split the *object-verb* (*obj*) dependency into subtypes as *direct object*, *indirect object*, and *object of preposition*. Although the *direct object* subtype achieved more successful results than the *obj* dependency, the improvement was not statistically significant. So, we continued the analysis with the original *obj* dependency. Table 3 shows the dependencies used in this work.

3.3. Datasets

In this study, we use three datasets from the UCI machine learning repository: Reuters-21578 (Reuters), National Science Foundation research award abstracts (NSF), and mini 20 newsgroups (MiniNg20) (Asuncion and Newman, 2007). We chose datasets with different characteristics in order to observe the effect of the methods on different types of data.

Reuters is a well-known formal dataset that has been used in many TC algorithms (Özgür et al., 2005; Yang and Liu, 1999). We use the standard Mod-Apte split in which there are 9603 training documents and 3299 test documents (Özgür et al., 2005). All the topics that exist in both the training set and the test set were utilized in the experiments. The dataset thus 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, indicating 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 (Asuncion and Newman, 2007). Year 2001 was selected randomly and five sections (four sections for training and one section for test) were picked out from this year. We formed five different splits, repeated all the tests with these five cross folds, and took their average as the final result.

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

3.4. System components

Fig. 1 shows the system architecture including the main components in the system. In this section, we explain these components and their roles in the overall architecture.

3.4.1. Preprocessing

The first step is the preprocessing of the datasets, where documents are parsed, non-alphabetic characters and mark-up tags are discarded, case-folding is performed, and stopwords (for word

Table 3
Dependency pattern types with their examples.

Symbol	Type	Example	Symbol	Type	Example
acomp	adjectival comp.	turn-bad	adv	adverbal cls. modifier	quickly-open
agent	agent	approve-bank	amod	adjectival mod.	scientific-study
app	appositional mod.	monitoring-detection	attr	attributive	remain-year
aux	auxiliary passive	expected-are	cls	clause modifier	use-determine
comp	complement	decline-disclose	complm	complementizer	is-that, have-that
conj	conjunctive	energy-chemical	infmod	infinitival mod.	way-invest
mark	mark	account-while	nn	noun compound mod.	source-laser
obj	object-verb	glass-break	part	participle mod.	costs-related
poss	possession mod.	Asia-nations	prep	prepositional mod.	focus-research
prep-along	along prep. mod.	moves-chromosomes	prep-as	as prep. mod.	farming-strategy
prep-at	at prep. mod.	available-institution	prep-btwn	between prep. mod.	relation-algebra
prep-by	by prep. mod.	displayed-species	prep-for	for prep. mod.	use-study
prep-from	from prep. mod.	show-studies	prep-in	in prep. mod.	low-cost
prep-into	into prep. mod.	extend-regions	prep-none	generic prep. mod.	clarify-by
prep-of	of prep. mod.	modeling-behavior	prep-on	on prep. mod.	work-project
prep-over	over prep. mod.	stayed-time	prep-to	to prep. mod.	similar-theory
prep-with	with prep. mod.	vary-depth	prt	phrasal verb participle	cover-up
rcmod	relative cls. mod.	begins-season	rel	relative mod.	begin-season
subj	subject-verb	they-break			

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

features) are eliminated. We use the list of 571 stopwords of the Smart system (Salton et al., 1975). Using a stoplist significantly reduces the feature vector size and the memory requirements of the system (Manning et al., 2008). 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 (Manning et al., 2008). 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, we chose the Porter stemmer which is one of the most experienced stemmers for word forms (Özgür and Güngör, 2009; Porter, 1980).

There are several approaches (tf-idf weighting, boolean weighting, Okapi BM25, etc.) for weighting the terms used in the machine learning algorithm. Boolean weighting is the simplest one but it is usually outperformed by tf-idf (Özgür et al., 2005; Salton and Buckley, 1988). Okapi BM25 is a non-binary model used mainly for query-document similarity, related search algorithms, and relevance feedback (Robertson et al., 2000). It takes into account the current document length, the average length of all the documents and the term frequencies, and attempts to tune two parameters empirically. In this work, our motivation is to compare the proposed approaches and improve the bow approach rather than analyzing different term weighting methods for text classification. So we chose the widely used and efficient tf-idf weighting to be used in all the proposed approaches.

We use the following standard form of tf-idf (Manning et al., 2008) to calculate the weight of a term t in a document d, where $tf_{t,d}$ is the frequency of term t in document d (each document vector is normalized to unit length to account for different document lengths), N is the total number of documents, and df_t is the number of documents in the dataset that include t:

$$\text{tf-idf} = tf_{t,d} * log\frac{N}{df_t} \tag{1}$$

3.4.2. Machine learning tool

As stated in Section 2.1, SVM with linear kernel was shown to yield the leading results in text classification, so we decided to use this classifier as the machine learning module. In our experiments, we used the SVM^{light} system which is an efficient implementation by Joachims (1999) and has been commonly used in previous studies. We use the one-versus-all mode for dataset topics for SVM classification (Forman, 2003).

3.4.3. Syntactic tool

Stanford parser is known to be one of the most powerful and efficient parsers having the least error rate (Stevenson and Greenwood, 2006). In our initial tests, we observed that it averts syntactic ambiguities in the sentences successfully and gives the first probable parse as the result. It is a statistical parser and has an integrated capability of extracting both the part-of-speech information and the dependencies between the words in a sentence. In this work, we use the Stanford parser to parse the sentences and obtain the lexical dependencies shown in Table 3. The PCFG parser mode was selected in our implementation (Klein and Manning, 2003).

4. Experiments and results

Based on the three approaches discussed in Section 3 (AW, AWP and AWDP), our first motivation in this section is to determine the optimal PL values in all the datasets. We then analyze in the third part of the section whether the proposed approaches with the determined PL values outperform the baseline AW method. In the fourth part, the results are analyzed from the dataset type perspective. We extend the experiment setup with the split of specific dependencies in the next subsection. Then, we perform statistical analysis in order to observe the significance of the improvements of the proposed approaches. Finally, we state the hardware specifications and time complexities.

4.1. Success measures

To evaluate the performance of the proposed approaches, we use the commonly used F-measure metric, which is equal to the harmonic mean of recall (ρ) and precision (π) (Manning et al., 2008). They are defined as follows:

$$\pi_i = \frac{TP_i}{TP_i + FP_i}, \quad \rho_i = \frac{TP_i}{TP_i + FN_i} \tag{2}$$

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 and FN_i (false negatives) is the number of documents that actually belong to class i but are not assigned 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 using two different types of averaging methods, namely micro-average and macro-average (Manning et al., 2008).

In micro-averaging, F-measure is computed globally without categorical discrimination. That is, all classification decisions in the entire dataset are taken into account when computing the F-measure score as shown below:

$$F(\text{micro-averaged}) = \frac{2 * \pi * \rho}{\pi + \rho} \tag{3}$$

where π and ρ denote, respectively, the precision and recall values over all the classification decisions. 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 performance of the classifier 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:

$$F_i = \frac{2 * \pi_i * \rho_i}{\pi_i + \rho_i}, \quad F(\text{macro-averaged}) = \frac{\sum_{i=1}^{M} F_i}{M} \tag{4}$$

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 performance of the classifier on rare categories. In the experiments, we provide both measurement scores to be more informative.

4.2. Pruning level analysis

In the experiments, we first applied the AW method in which the feature vector consists of all the words in the dataset without any pruning. Then the AWP method was applied with different pruning levels for words. Among the pruning levels used, the best results (high accuracies with minimum feature numbers) were obtained around PL = 13 in all the three datasets as can be seen in Table 4. In the table, Feature# represents the number of features. The datasets used in this work are independent by having different properties (formal/informal, skewed/balanced, etc.) and the two success measures used have different characteristics (MicroF is

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

Table 4
Feature numbers and success rates in different word pruning levels (AWP) (numbers in bold indicate the results for the optimal PL value).

	Reuters			NSF			MiniNg20		
PL	Feature#	MicroF	MacroF	Feature#	MicroF	MacroF	Feature#	MicroF	MacroF
1	20292	85.58	43.83	13424	64.46	46.11	30970	46.42	43.44
2	12959	85.55	43.84	8492	64.41	46.21	13102	49.73	47.13
3	9971	85.52	43.93	6328	64.62	46.42	9092	49.64	47.19
5	7168	85.51	44.56	4528	64.86	46.49	6000	51.26	48.52
8	5268	85.73	44.91	3376	64.66	46.38	4169	52.48	49.90
13	**3976**	**85.84**	**44.85**	**2478**	**64.58**	**46.49**	**2863**	**53.62**	**51.02**
20	3046	86.02	44.55	1875	64.23	46.67	2025	53.78	51.02
30	2237	81.29	43.59	1419	63.84	46.21	1384	52.89	50.46

document-based while MacroF is class-based). The number of all documents in these datasets ranges between 2000 and 12,902, with respectively 1600 and 9603 training documents. Although PL = 13 seems as the optimal pruning level in such datasets, we also analyzed the effect of the dataset size on pruning by repeating the experiments with varying sized subsets (200, 500, 1000, etc. training documents) of all the three datasets. We observed that, as can be expected, the optimal pruning level decreases as the number of documents decreases. However, the optimal values were always above PL = 2 even with the smallest subset containing 200 training documents. Based on these results, we can claim that pruning is a necessary preprocessing step, it should be implemented using values higher than PL = 2 even for very small datasets (in contrary to the works in the literature which usually fix the pruning level to some arbitrary small value such as two), and for medium size standard datasets containing about 2000–13,000 documents PL values between 10 and 15 yield the best results for text classification.

For the AWDP method, we fixed the word PL value to 13 and repeated the experiments in the standard datasets for each of the 37 dependencies and different dependency pruning levels. As mentioned in Section 2.2, a dependency is formed by combining two dependent words, so both of the words must be repeated in the same pattern for the dependency to reoccur in the dataset. This characteristic makes the dependency analysis different from words: a large number of dependencies but mostly with low frequencies. Due to this difference, we decided to perform an independent pruning level analysis for dependencies.

Fig. 2 shows the performances of the methods as a function of dependency PL values. The black colored parts of the bars in the figure correspond to the success rates of the AW method (the success rates with PL = 1 in Table 4). The gray colored part shows, when compared with the AW method, the increment in the success rate of the AWP method with the optimal word PL value (the success rates with PL = 13 in Table 4). We see that AWP with this optimal pruning value outperforms AW for all the datasets. The white colored part is the success increment obtained by the AWDP method for different pruning levels of dependencies. To give a general idea about the effect of using the successful dependencies in classification, we show the AWDP results in the figure by taking the average of the leading three dependencies for each dataset (e.g. *prep-in*, *prep-from*, and *amod* for Reuters – see Table 5). As will be discussed in the next section, different dependency types have different effects on the performance. Our goal in this research is analyzing each dependency type independent of others and identifying those that increase the performance of the classifier.

In the formal datasets Reuters and NSF, the success rates follow a similar pattern: the MicroF score decreases with increasing (dependency) PL value and increases a little for PL value 5 or 8, whereas the MacroF value first increases up to PL value 5–8 and then decreases. The situation is somewhat different for informal MiniNg20, where both scores increase with PL = 2 and then remain almost stable up to PL = 20. The MiniNg20 dataset seems less sensitive to dependency pruning. Taking into account the increase in the success rates as well as the decrease in the size of the feature vector (Table 6), we identified the best dependency pruning levels as 8 for Reuters and NSF, and as 2 for MiniNg20.

We can see from the figure that the pruning process almost always improves the success rate of the classifier. AWP outperforms AW by eliminating rare words in the feature vector and AWDP where dependencies with very low occurrences are ignored is more successful than AWP.

4.3. Comparison of the approaches

As stated in Section 3.1, one of the motivations of this study is to reach the same or better success rates with less features. In this section, we analyze the proposed approaches from the perspectives of success rates and feature numbers, which are the main criteria for classification performance, for words and dependencies. Feature number is simply the size of the solution vector used by the machine learning algorithm and the success rates are measured in terms of the commonly used MicroF and MacroF measures.

4.3.1. Success rates

Table 5 shows the AWDP classification results for the most successful 10 dependencies in each dataset. The table also includes the AW and AWP scores for comparison. The PL symbol in the table denotes the pruning level for words (first number) and dependencies (second number). As stated previously, the AWDP method makes use of the instances of a single dependency in addition to the words. For instance, the entry *prep-in* for Reuters in the table denotes the result of the experiment where the feature vector was formed of the dependencies *prep-in* only (with PL = 8) and the words (with PL = 13).

As can be seen from the table, there are common dependency types (shown in bold) in the 10 most successful dependency lists of the datasets. In Reuters and NSF, four of the 10 best dependencies (among 37 dependency types) are the same. However, MiniNg20 has only one common dependency with NSF and Reuters. This is probably due to the writing style in the datasets: the formal datasets Reuters and NSF include mostly grammatical sentences, whereas the informal MiniNg20 contains many ungrammatical sentences, partial phrases, and spelling errors. These dataset specific differences will be discussed in Section 4.4.

The table shows that AWP is more successful than AW by about 0.20–0.30% in Reuters and NSF, and by about 7–8% in MiniNg20. In addition, AWDP with leading dependencies improve the AWP scores by about 0.10–0.20% (MicroF and MacroF) in Reuters, while the improvement is about 0.50–0.60% in NSF and MiniNg20. We can conclude that both types of pruning (word and dependency) contribute to the success rates.

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

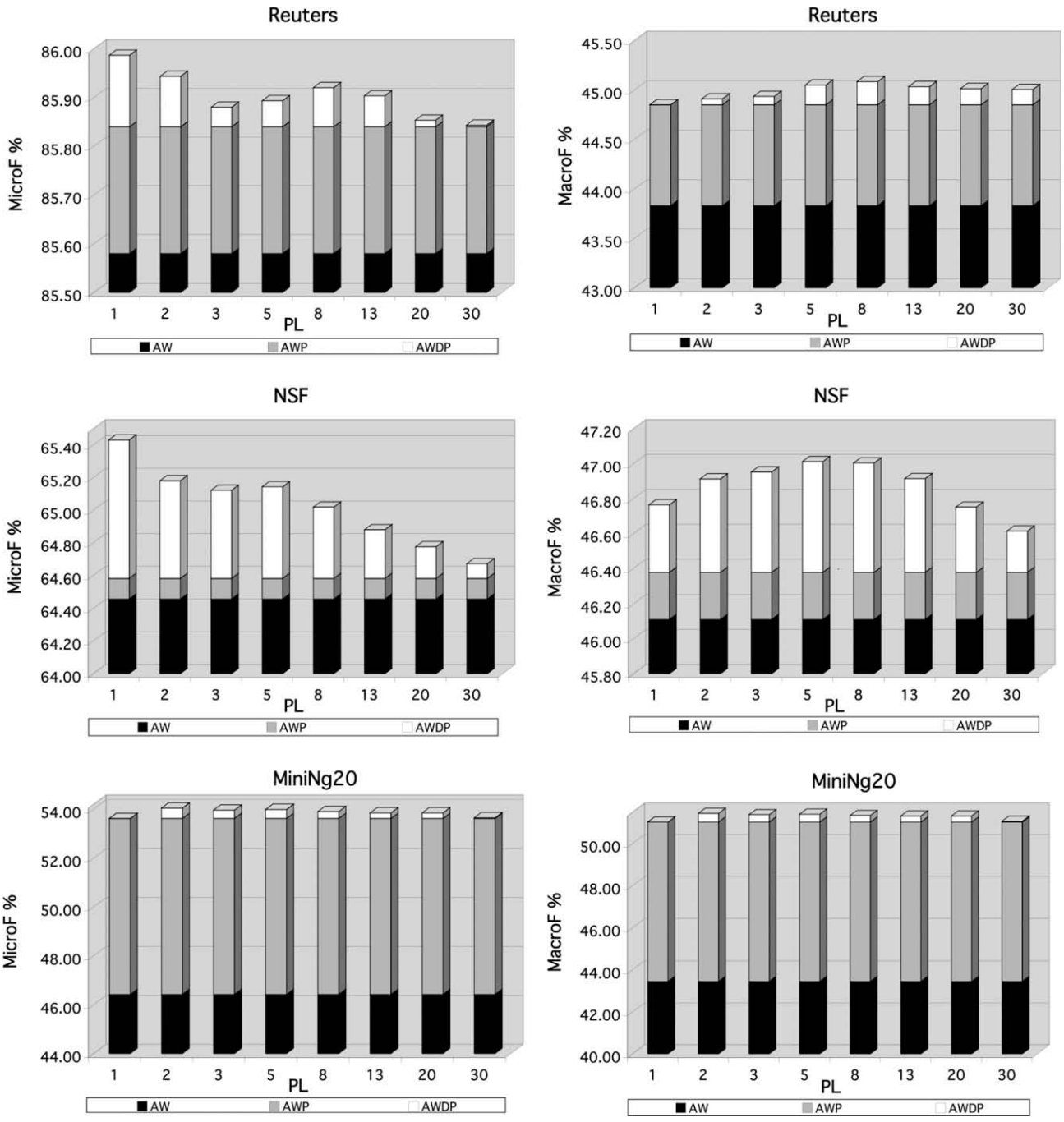

Fig. 2. Success rates of AW, AWP, and AWDP for Reuters, NSF, and MiniNg20.

4.3.2. Feature numbers

Tables 4 and 6 show, respectively, the number of word features and the number of dependency features (for the most successful dependency in each dataset) included in the feature vector at different pruning levels. PL = 1 indicates the total number of words and dependencies (for the selected dependency type) in the related document collection. When the PL value is increased by 1 (PL = 2), 40–60% of the words and 75–80% of the dependencies are eliminated, which indicates that the feature vectors are mostly sparse when there is no pruning. As the pruning level increases more, the effect of pruning diminishes. For instance, increasing the word PL from 20 to 30 eliminates only about 3% of all the words in the dataset. The bold numbers in the tables denote the number of elements in the feature vector under the optimal PL values. We see that the number of features with these pruning levels are similar in all the experimented datasets: 2500–4000 words and 220–350 dependencies. In related studies, it was reported that about 2000 words (without dependency usage) yield the best success rates (Özgür et al., 2005), which is consistent with our results. By appending an additional 220–350 dependencies (about 10% of the word features) to the feature vector containing 2500–4000 words, we achieve a significant improvement in the success rates. Similar to the change in the optimal PL value depending on the size of the dataset as explained in Section 4.2, the optimal feature number also tends to diminish when smaller datasets (e.g. subsets of the datasets with 200, 500, 1000, etc. training documents) are used. However, for medium size standard datasets, we observe a consistent behavior as mentioned above.

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

Table 5
Leading dependencies in AWDP method for Reuters, NSF, and MiniNg20.

	Reuters	PL: 13–8		NSF	PL: 13–8		MiniNg20	PL: 13–2	
		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
6	prep-to	85.91	44.92	prep	64.81	46.73	cls	53.87	51.21
7	**adv**	85.84	44.96	**adv**	64.66	46.72	**prep-as**	53.87	51.21
8	prep-with	85.86	44.94	prep-of	64.61	46.65	prep-at	53.87	51.21
9	**obj**	85.91	44.88	**prep-as**	64.63	46.61	prep-by	53.87	51.21
10	**app**	85.80	44.98	conj	64.65	46.57	prep-on	53.87	51.21
	AWP	85.84	44.85	AWP	64.58	46.49	AWP	53.62	51.02
	AW	85.58	43.83	AW	64.46	46.11	AW	46.42	43.44

Table 6
Number of dependencies in different dependency pruning levels (AWDP) (numbers in bold indicate the results for the optimal PL value).

PL	Reuters prep-in	NSF nn	MiniNg20 prt
1	11,792	22,828	907
2	2805	4734	**251**
3	1147	1953	126
5	467	759	49
8	**222**	**351**	16
13	93	146	8
20	57	61	6
30	37	29	2

4.4. Dataset comparison

The datasets used in this work can be classified into two categories according to their formality levels: Reuters and NSF have a formal style, whereas MiniNg20 is mostly informal. Since the efficiency of parsing is related to the grammaticality of sentences in a document, MiniNg20 leads to less accurate parse results due to morphological and syntactic errors. This is evidenced in the pruning process; about 60% of the words and 70% of the dependencies are eliminated with PL = 2, and the optimal dependency PL value is limited to 2 (which is 8 for the other datasets) because of the high pruning rate. In addition, the pruning process increased the success rates in MiniNg20 much more than Reuters and NSF (an increase of about 8% between AW and AWDP). This result shows the success of pruning especially in informal datasets.

As mentioned previously, Reuters and NSF have some common dependencies in the list of successful dependencies, but this is not the case for MiniNg20. For instance, the *comp* dependency (a structurally complicated dependency formed by integrating two verbs that have the same subject in adjacent clauses) gives successful results in formal datasets. However, in informal datasets, it does not improve the performance due to the simple or ungrammatical sentence structure. Instead of such dependencies, *prt* (phrasal verb participle) is one of the simplest dependencies which yields the most successful results with MiniNg20.

Another criterion that affects the success rates is dataset skewness. As the skewness of a dataset increases, the gap between the MicroF and MacroF values enlarges. This is due to the fact that in skewed datasets we do not have available sufficient number of documents in some classes and this causes a decrease in the class-based MacroF metric. Reuters is a highly skewed dataset; NSF is also skewed but less than Reuters. Table 5 shows that the ratio of MicroF and MacroF scores is about 1.9 and 1.4 in Reuters and NSF, respectively. On the other hand, MiniNg20 is a balanced dataset and gets similar MicroF and MacroF values.

4.5. Split of prepositions

As stated in Section 3.2, we split the *prep* dependency into 15 possible subdependencies due to the unsuccessful results obtained with the combined form (Özgür and Güngör, 2009). We observed from the experiments that these more specific prepositional dependency types have more discriminative power than the generic *prep* dependency and increase the success rates. Table 5 shows that 10 of these dependencies are among the overall 30 most successful dependency patterns of the datasets. However, we cannot observe a prepositional dependency type common to all the datasets; only the *prep-as* dependency occurs in the list of successful dependencies in more than one dataset. We conclude that making the *prep* dependency more specific improves the classification performance, but the improvement is obtained with different subtypes in different datasets.

4.6. Statistical analysis of success rates

We used the standard sign test to measure the significance of the improvements in the proposed system. In this significance test, two systems are compared based on their binary decisions on all the document/topic pairs. Binary decision states whether a document belongs to that topic or not. The correctness of the decisions are compared for each instance (Yang and Liu, 1999). Standard z values, which represent the number of standard deviations a given value x falls from the mean μ, are calculated for each comparison. For each z value, the corresponding confidence levels (probability that the interval estimate contains the population parameter; in our case the population parameter is the superiority of one method over the other one in the comparison of the correctness of their decisions) are determined according to the standard normal distribution (Larson and Farber, 2000; Montgomery, 2001). Table 7 shows the comparison of the three methods: AW, AWP with optimal word PL values (13 for all the datasets), and AWDP with optimal word PL values, optimal dependency PL values (8 for Reuters and NSF, 2 for MiniNg20) and best dependency types (*prep-in* for Reuters, *nn* for NSF, *prt* for MiniNg20).

The proposed method (AWDP) is better than the standard AW approach with 95.82%, 88.69%, and 99.95% confidence levels (conf.) in Reuters, NSF, and MiniNg20, respectively. When we compare AWP and AWDP, we see that the use of dependencies for classification in addition to pruning also improves the results significantly for Reuters and NSF. When we combine the results of all the datasets as shown in the last column of the table, we see that

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

Table 7
Statistical comparison of the approaches.

Comparisons	Reuters		NSF		MiniNg20		Overall	
	z	conf. %	z	conf. %	z	conf. %	z	conf. %
AWDP over AWP	1.46	92.78	3.40	99.97	0.25	59.87	3.82	**100.00**
AWDP over AW	1.73	95.82	1.21	88.69	3.29	99.95	2.73	**99.68**

AWDP significantly outperforms both the AW and the AWP methods.

4.7. Hardware specifications and time complexities

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

For AWDP, dataset parsing is the most time consuming part of the overall process and takes more than 10 h for all the datasets. However, the parsing operation is performed only once before all the experiments on the dataset. AW and AWP do not involve any parsing module. For these methods, creating the tf-idf values for the words in the dataset during the training and test phases consumes the most time. This process takes approximately 10 min with about 10,000 features.

5. Conclusions

In this study, we proposed the use of the lexical dependency and pruning concepts for text classification as an extension to the standard bow approach. To the best of our knowledge, this is the first study that makes a detailed analysis of the effect of 37 different dependencies and eight different pruning levels in the text classification domain. We have shown that both of the approaches significantly improve the results of the standard bow approach, by also reducing the dimensionality of the feature vector. Observing the effect of each dependency pattern separately and using the most effective ones under the optimal pruning levels improves the perspective of the bow approach by compensating for the lack of ignoring the relations between the words in the standard algorithm.

Word pruning was used in several previous studies, but the pruning level was fixed to some value (usually a small value such as two) in these studies without any further analysis (Nastase et al., 2006). In this work, we determined that the optimal pruning value is usually much higher and it is possible to improve the performances of classical TC algorithms by a correct choice of the PL value. For all the datasets used in this research with different characteristics, the optimal pruning level for words was found to be around 13 yielding about 2500–4000 keywords. We also analyzed the effect of the dataset size on pruning by using smaller subsets of these datasets. Based on all these results, we claim that pruning is a necessary preprocessing step, it should be implemented using values higher than PL = 2 even for very small datasets, and for medium size standard datasets with around 2000–13,000 documents, PL values between 10 and 15 yield the best results for text classification.

In addition to words, we analyzed the effect of including lexical dependencies in the feature vector. We analyzed 37 different dependency types separately and identified the most discriminative ones for each dataset with the optimal pruning levels (8 for the Reuters and NSF datasets and 2 for the MiniNg20 dataset).

The pruning process consistently improved the success rate of the classifier in the experiments. AWP outperformed AW by eliminating rare words in the feature vector and AWDP yielded significantly more successful results than AWP by using pruned dependencies in the solution vector. The optimal number of word features was found to be in the range of 2500–4000 and appending an additional 220–350 dependencies to the feature vector increased the success rates significantly.

By analyzing the results from the dataset perspective, we observed that the formality level of a dataset is an important factor and the parameters of the classifier (pruning levels, dependency types, etc.) should be set accordingly. The formal Reuters and NSF datasets had three common dependencies in the list of successful dependencies, but this was not the case for MiniNg20. For instance, the *comp* dependency gave successful results in formal datasets. However in informal datasets, instead of this complex dependency, one of the simplest dependencies (*prt*) yielded the best result due to the simple and ungrammatical sentence structure. Also, the initial pruning level caused much more feature filtering and increase in success rate in informal MiniNg20 when compared with the formal datasets.

As future work, we plan to combine the successful dependencies in a dataset and include them in the feature vector as an extension to the proposed system. We will analyze the PL values of the dependencies in order to find the optimal combination. This type of additional information will probably yield better performance in terms of accuracy and time. We also work on integrating the space reduction metrics mentioned in Section 3.1 (tf-idf, mutual information, LSI, etc.) with the pruning implementation and dependency usage for the text classification problem. Another possible extension is repeating the tests in more datasets with different formality levels and skewness properties, so that we can develop robust algorithms for automatic detection of useful dependencies according to dataset properties.

Acknowledgements

This work has been supported by the Boğaziçi University Research Fund under the Grant No. 05A103D.

References

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

Basili, R., Pazienza, M.T., Mazzucchelli, L., 2000. An adaptive and distributed framework for advanced IR. In: Proc. Internat. Conf. on Adaptivity, Personalization and Fusion of Heterogeneous Information (RIAO 2000), pp. 908–922.

Cahill, A., Heid, U., Rohrer, C., Weller, M., 2009. Using tri-lexical dependencies in LFG parse disambiguation. In: The 14th Internat. LFG Conf., Cambridge.

Charniak, E, Knight, K., Yamada, K., 2003. Syntax-based language models for statistical machine translation. In: Proc. MT Summit IX. Internat. Association for Machine Translation.

Forman, G., 2003. An extensive empirical study of feature selection metrics for text classification. J. Machine Learn. Res. 3, 1289–1305.

Furnkranz, J., Mitchell, T., Rilof, E., 1998. A case study in using linguistic phrases for text categorization on the WWW. In: AAAI-98 Workshop on Learning for Text Categorization.

Ghanem, M., Guo, Y., Lodhi, H., Zhang, Y., 2002. Automatic scientific text classification using local patterns. ACM SIGKDD Explor. Newsl. 4 (2), 95–96.

Herrera, J., Penas, A., Verdejo, F., 2006. Textual entailment recognition based on dependency analysis and WordNet. In: Proc. PASCAL Challenges Workshop on Recognising Textual Entailment. Lecture Notes in Computer Science, vol. 3944. Springer, Berlin, Heidelberg.

Joachims, T., 1999. Advances in Kernel Methods – Support Vector Learning. MIT Press.

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005

Klein, D., Manning, C., 2003. Fast exact inference with a factored model for natural language parsing. Advances in Neural Information Processing Systems, vol. 15. MIT Press, Cambridge.

König, A.C., Brill., E., 2006. Reducing the human overhead in text categorization. In: Proc. 12th ACM SIGKDD Internat. Conf. Knowledge Discovery and Data Mining, Philadelphia, USA, pp. 598–603.

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

Lewis, D.D., 1992. An evaluation of phrasal and clustered representations on a text categorization task. In: Proc. 15th Annual Internat. ACM SIGIR Conf. on Research and Development in Information Retrieval, Copenhagen, Denmark, pp. 37–50.

Liu, T., Chen, Z., Zhang, B., Ma, W., Wu, G., 2004. Improving text classification using local latent semantic indexing. In: Proc. Fourth IEEE Internat. Conf. on Data Mining.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Christianini, N., Watkins, C., 2002. Text classification using string kernels. J. Machine Learn. Res. 2, 419–444.

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

Marneffe, M.C., MacCartney, B., Manning, C., 2006. Generating typed dependency parses from phrase structure parses. In: Proc. Internat. Conf. on Language Resources and Evaluation (LREC 2006), Genoa, Italy, pp. 449–454.

Montgomery, Douglas C., 2001. Design and Analysis of Experiments. John Wiley.

Moschitti, A., 2008. Kernel methods, syntax and semantics for relational text categorization. In: Proc. ACM 17th Conf. on Information and Knowledge Management (CIKM), Napa Valley, California.

Moschitti, A., Basili, R., 2004. Complex linguistic features for text classification. In: Proc. European Conf. on Information Retrieval (ECIR 2004), pp. 181–196.

Mullen, T., Collier, N., 2004. Sentiment analysis using support vector machines with diverse information sources. In: Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP 2004).

Nastase, V., Shirabad, J.S., Caropreso, M.F., 2006. Using dependency relations for text classification. In: Proc. Nineteenth Canadian Conf. on Artificial Intelligence, Quebec, Canada.

Özgür, A., Özgür, L., Güngör, T., 2005. Text categorization with class-based and corpus-based keyword selection. In: Proc. 20th Internat. Symposium on Computer and Information Sciences (ISCIS 2005). Lecture Notes in Computer Science, vol. 3733. Springer-Verlag, Berlin, Heidelberg, pp. 606–615.

Özgür, L., Güngör, T., 2009. Analysis of stemming alternatives and dependency pattern support in text classification. In: Proc. Tenth Internat. Conf. on Intelligent Text Processing and Computational Linguistics (CICLing 2009), Mexico City.

Porter, M., 1980. An algorithm for suffix stripping. Program 14 (3), 130–137.

Robertson, S.E., Walker, S., Beaulieu, M., 2000. Experimentation as a way of life: Okapi at TREC. Inform. Process. Manage. 36, 95–108.

Salton, G., Buckley, C., 1988. Term weighting approaches in automatic text retrieval. Inform. Process. Manage. 24 (5), 513–523.

Salton, G., Yang, C.S., Wong, A., 1975. A vector-space model for automatic indexing. Commun. ACM 18 (11), 613–620.

Stevenson, M., Greenwood, M., 2005. A semantic approach to IE pattern induction. In: Proc. 43rd Annual Meeting of the ACL, Ann Arbor.

Stevenson, M., Greenwood, M., 2006. Comparing information extraction pattern models. In: Proc. Workshop on Information Extraction Beyond the Document, Sydney, pp. 12–19.

Wang, Z., Zhang, D., 2006. Feature selection in text classification via SVM and LSI. In: Proc. Third Internat. Symposium on Neural Networks (ISNN 2006). Springer-Verlag, pp. 1381–1386.

Wellner, B., Pustejovsky, J.D., Havasi, C., Rumshisky, A., Sauri, R., 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In: Proc. SIGdial Workshop On Discourse and Dialogue, Sydney, pp.117–125.

Yang, Y., Liu, X., 1999. A re-examination of text categorization methods. In: Proc. 22nd ACM SIGIR Conf. on Research and Development in Information Retrieval, Berkeley, pp. 42–49.

Please cite this article in press as: Özgür, L., Güngör, T. Text classification with the support of pruned dependency patterns. Pattern Recognition Lett. (2010), doi:10.1016/j.patrec.2010.05.005