A Case Study in Using Linguistic Phrases for Text Categorization on the WWW

Johannes Fürnkranz
juffi@cs.cmu.edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Tom Mitchell
mitchell+@cs.cmu.edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Ellen Riloff
riloff@cs.utah.edu
Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Abstract

Most learning algorithms that are applied to text categorization problems rely on a *bag-of-words* document representation, i.e., each word occurring in the document is considered as a separate feature. In this paper, we investigate the use of linguistic phrases as input features for text categorization problems. These features are based on information extraction patterns that are generated and used by the AutoSlog-TS system. We present experimental results on using such features as background knowledge for two machine learning algorithms on a classification task on the WWW. The results show that phrasal features can improve the precision of learned theories at the expense of coverage.

Introduction

Most machine learning algorithms for text categorization represent documents as a *bag of words*. Typically, each word is treated as a separate feature. For example, a HTML-document might be classified as the home page of a student if the word *student* occurs on the page. However, *student* will also occur on many other pages on the WWW, including most pages of computer science departments. On the other hand, a sentence like *"I am a student of computer science at Carnegie Mellon University"* is clearly indicative of a student's home page. It is worth noting that none of the words that appear in this sentence are good predictors of a student home page, nor is any subset of these words, because all of them are quite common on pages related to Carnegie Mellon's School of Computer Science.[1] What makes a difference is the linguistic role that these words have in this sentence.

AutoSlog-TS, originally conceived for information extraction (Riloff 1996a), is able to provide a learner with features that capture some of the syntactic structure of natural language text. For the above input sentence, it extracts the following four features: *I am* <_>, <_> *is student*, *student of* <_>, and *student at* <_>.[2] All four of them are highly indicative of student pages and at least the last three of them are quite unlikely to appear on other types of pages. Note that the last of these, *student at* <_>, does not match a contiguous piece of text, but is based on prepositional attachment to the word *student*.

In previous work (e.g., (Riloff & Lorenzen 1998)), promising results on the usefulness of such phrases for text categorization tasks were obtained using simple statistical thresholding methods to find the best classification terms. In this paper, we report on some experiments that aimed at investigating the utility of such phrases as features for two state-of-the-art machine learning algorithms, namely the Naive Bayes classifier Rainbow and the separate-and-conquer rule learning algorithm Ripper. The case study is performed in a text classification task on the WWW.

Generating Phrasal Features

AutoSlog was developed as a method for automatically constructing domain-specific extraction patterns from an annotated training corpus. As input, AutoSlog requires a set of noun phrases, i.e., the information that should be extracted from the training documents. AutoSlog then uses syntactic heuristics to create linguistic patterns that can extract the desired information from the training documents (and from unseen documents). The extracted patterns typically represent subject–verb or verb–direct-object relationships (e.g., <*subject*> *teaches* or *teaches* <*direct-object*>) as well as prepositional phrase attachments (e.g., *teaches at* <*noun-phrase*> or *teacher at* <*noun-phrase*>). See (Riloff 1996b) for a more detailed description of AutoSlog.

The key difference between AutoSlog and AutoSlog-TS (Riloff 1996a) is the removal of the need for an annotated training corpus. AutoSlog-TS simply generates extraction patterns for *all* noun phrases in the training corpus whose syntactic role matches one of the heuristics. Table 1 shows three of the 16 syntactic heuristics employed in the current

[1] In this sentence, the most predictive words for a student home page are *I* and *am*. They are quite reliable features for the problem of discriminating between student pages and other departmental pages and, for this task, should not be removed by a stop list (see also (Craven *et al.* 1998a)).

[2] We use angle brackets <> to represent variables.

Syntactic Heuristic	Phrasal Feature
noun aux-verb <d-obj>	*I am <_>*
<subj> aux-verb noun	*<_> is student*
noun prep <noun-phrase>	*student of <_>*
	student at <_>

Table 1: Syntactic heuristics that are used for finding phrasal features in the sentence *"I am a student of computer science at Carnegie Mellon University."*

version of AutoSlog-TS as well as their corresponding extraction patterns that can be detected in the example sentence *"I am a student of computer science at Carnegie Mellon University"*. Note that the third syntactic heuristic of table 1 fires twice in the same sentence using two different prepositional attachments to the noun *student*. Also, the different forms of the verb *to be* are not discerned for the first two patterns. However, in general words that occur in different forms will not be treated in the same way. For example, the sentence *"Here are the students at our department"* will not match the last extraction pattern of table 1, because the word *student* occurs in plural instead of singular. It seems likely that the plural version *students at <_>* occurs more frequently on departmental pages than on student home pages. That such small differences in the use of words can make a big difference for text classification was previously observed in (Riloff 1995). A set of experiments demonstrated that the occurrence or absence of linguistic phrases of the above form can be successfully used for recognizing relevant documents of the terrorist domain of the *4th Message Understanding Conference (MUC-4)* (Riloff & Lorenzen 1998).

In this paper, we explore the potential use of the extraction patterns generated by AutoSlog-TS as phrasal features for state-of-the-art learning algorithms. In the following, we will briefly describe the learning algorithms and the test domain (classifying WWW pages related to computer science departments), and then discuss the results of our experiments.

Description of the Experiments

We evaluated the use of phrasal features with two different learning algorithms, the naive Bayes classifier Rainbow and the rule learning algorithm Ripper. The data set used for our experiments is the *4-universities* dataset, which has been collected for the WebKB project at Carnegie Mellon University.

Rainbow

Rainbow is a Naive Bayes classifier for text classification tasks (Mitchell 1997), developed by Andrew McCallum at CMU[3]. It estimates the probability that a document is a member of a certain class using the probabilities of words occurring in documents of that class independent of their context. By doing so, Rainbow makes the naive independence assumption. More precisely, the probability of document d belonging to class C is estimated by multiplying the prior probability $\Pr(C)$ of class C with the product of the probabilities $\Pr(w_i|C)$ that the word w_i occurs in documents of this class. This product is then normalized by the product of the prior probabilities $\Pr(w_i)$ of all words.

$$\Pr(C|d) := \Pr(C)\prod_{i=1}^{n}\frac{\Pr(w_i|C)}{\Pr(w_i)} \qquad (1)$$

As many of the probabilities $\Pr(w_i|C)$ are typically 0.0 (hence their product will be 0.0), Rainbow smoothes the estimates using the technique proposed by Witten & Bell (1991). A related problem — the fact that for text classification tasks many estimates of $\Pr(C|d)$ for the winning class tend to be close to 1.0 and often will be exactly 1.0 because of floating-point round-off errors — is addressed by providing an option to incorporate a correction term based on Kullback-Leibler Divergence. This correction does not change the classification of the documents, but provides more realistic probability estimates. This option was used in our experiments to obtain better confidence estimates for the predictions, which we used for generating recall/precision graphs. A more detailed description of this smoothing technique and of Rainbow in general can be found in (Craven *et al.* 1998a).

Ripper

Ripper[4] (Cohen 1995) is an efficient, noise-tolerant rule learning algorithm based on the incremental reduced error pruning algorithm (Fürnkranz & Widmer 1994; Fürnkranz 1997). It learns single rules by greedily adding one condition at a time (using Foil's information gain heuristic (Quinlan 1990)) until the rule no longer makes incorrect predictions on the growing set, a randomly chosen subset of the training set. Thereafter, the learned rule is simplified by deleting conditions as long as the performance of the rule does not decrease on the remaining set of examples (the pruning set). All examples covered by the resulting rule are then removed from the training set and a new rule is learned in the same way until all examples are covered by at least one rule. Thus, Ripper is a member of the family of separate-and-conquer (or covering) rule learning algorithms (Fürnkranz 1998).

What makes Ripper particularly well-suited for text categorization problems is its ability to use *set-valued features* (Cohen 1996). For conventional machine learning algorithms, a document is typically represented as a set of binary features, each encoding the presence or absence of a particular word in that document. This results in a very inefficient encoding of the training examples because much space is wasted for specifying the

[3]Available from `http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html`.

[4]Available from `http://www.research.att.com/~wcohen/ripperd.html`.

Class	All Examples		Test Examples	
Student	1641	(19.82%)	558	(13.42%)
Faculty	1124	(13.57%)	153	(3.68%)
Staff	137	(1.65%)	46	(1.11%)
Project	504	(6.09%)	86	(2.07%)
Department	182	(2.20%)	4	(0.10%)
Course	930	(11.23%)	244	(5.87%)
Other	3764	(45.45%)	3067	(73.76%)
Total	8282	(100.00%)	4158	(100.00%)

Table 2: Class Distribution in the 4 Universities data set

absence of words in a document. RIPPER allows to represent a document as a single set-valued feature that simply lists all the words occurring in the text. Conceptually, RIPPER's use of such a set-valued feature is no different than the use of binary features in conventional learning algorithms, although RIPPER makes use of some optimizations. For the remainder of this paper, we will continue to think of each word (or phrase) as a separate binary feature.

The WebKB Project

The goal of the WebKB Project (Craven *et al.* 1998b) is to extract a computer-understandable knowledge base from the WWW whose contents mirrors the contents of the WWW. Many applications could be imagined for such a knowledge base. For example, it could enhance the capabilities of currently available search engines that can only answer word-occurrence queries (e.g., ALTAVISTA) or that rely on a manually constructed knowledge base about the contents of WWW pages (e.g., YAHOO) by enabling them to answer questions like "Who teaches course X at university Y?" or "How many students are in department Z?". Currently, a prototype system uses an ontology of common entities in computer science departments (students, faculty, staff, courses, projects, departments) and relations between them (e.g., professor X is the instructor of course Y and the advisor of student Z). The prototype crawls the net and uses learned knowledge to classify pages of computer science departments into that ontology.

In order to furnish the crawler with some learned domain knowledge, a set of 8,282 training documents was collected from the WWW pages of various computer science departments.[5] About half of the pages are a fairly exhaustive set of pages from the computer science departments of four universities: Cornell, Texas, Wisconsin, and Washington. The remaining 4,120 pages are more or less randomly collected pages from various computer science departments. The pages are manually classified into the categories *Student*, *Faculty*, *Staff*, *Course*, *Project*, *Department*, and *Other*. Table 2 shows the frequency distributions of these classes in the data set. The *Other* class is a very heterogeneous class that contains pages found at these departments that are not classified as any of the six relevant classes. Note, however, that one of the assumptions made in manually classifying the pages was that each real-world entity is represented by only one page. Thus, if, e.g., a faculty member organizes his personal home page as a hypertext document containing separate pages for his research interests, his publications, his CV, and pointers to the research projects he is involved in, only the top page that links these informations together would be used to represent him in the class *Faculty*, while all other pages would be classified as *Other*. This is clearly not a perfect solution, as other people might organize the same information into a single page. Thus it can be expected to be hard to discriminate between certain pages of the *Other* class and pages of the relevant classes.

A more detailed description of this data set and of some previous work can be found in (Craven *et al.* 1998b; 1998a).

Experimental Setup

For our experiments, all pages were stripped of their HTML tags, converted to lower case, and all digits were replaced with a `D`. AUTOSLOG-TS was run on each document and the generated extraction patterns were encoded as one-word tokens and put into a separate file. We compared three different representations for each algorithm: One where each word was treated as a feature, one where each phrase (extraction pattern) was treated as a feature, and one where both were considered as features. For the last case, corresponding files were simply appended to produce the input files for RAINBOW, while for RIPPER we encoded each document with two set-valued features, one containing the words in the document, the other containing the tokens that represent the discovered phrasal features.

The algorithms were evaluated using the same procedure as in (Craven *et al.* 1998b): Each experiment consists of four runs, in which the pages of one of the four universities are left out in turn. Thus, each training set consists of the 4,120 pages from miscellaneous universities plus the pages from three of the four universities. The results of each of the four runs were combined using *micro-averaging*, i.e., the predictions made for the four test sets were thrown together and all evaluation measures were computed on this combined set. The frequency distribution of the classes in this combined test set is shown in the second column of table 2.

Unless noted otherwise, RIPPER was used with its default options. It should be noted that in this setting, RIPPER sorts the classes according to their inverse frequency and learns decision lists for discriminating one class against all classes ranked below it. Hence, the biggest class, in our case the *Other* class, is treated as a default class and no rules are learned for it. In the experiments with RAINBOW, we made use of its built-in stemming, and features that occurred only once were

[5]Available from `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`

Representation	RAINBOW	RIPPER
words	45.70%	77.78%
phrases	51.22%	74.51%
words+phrases	46.79%	77.10%

Table 3: Overall predictive accuracy

not considered. The experiments with RIPPER were performed without stemming and without any form of feature subset selection.

Results

Accuracy

Table 3 shows the results in terms of predictive accuracy on the 4 test sets combined. The results are quite diverse. For RAINBOW, the use of phrases increases predictive accuracy, while for RIPPER, the opposite is the case. An investigation of the confusion matrices (not shown) shows that this is mostly due to an increase in the number of pages that are classified as *Other*. For example, about 4.5% of the pages do not contain any natural language phrases and are classified as *Other* by default. This decrease in the number of pages for which a commitment to one of the six relevant classes is made, in some sense, confirms that the phrasal features are much more conservative in their predictions: They tend to appear less frequently, but some of them are highly typical for certain classes. This has a positive effect on the overall accuracy, because RAINBOW in general overpredicts the 6 minority classes. RAINBOW using word-based features classifies only 1216 pages as *Other*, while the test set contains 3067 pages of this class. RAINBOW using phrasal features classifies 1578 examples as *Other* without significantly changing the classifier's precision in the six relevant classes. This means that the majority of *Other* pages is misclassified into one of the six relevant classes, thus producing a low overall accuracy and a low precision, as we will see in the next section.

The situation is converse for RIPPER, which classifies 3008 pages as *Other* using words, while classifying 3517 pages as *Other* when using only phrases. Hence, for RIPPER, whose classification accuracy is above the default accuracy (73.76%), predicting more examples as *Other* has a negative effect on its performance.

These differences in the number of pages classified with the default class are also the main reason for the huge performance differences of the two algorithms. One reason for this phenomenon might be the nature of of the *Other* class, which is likely to contain many pages that are very similar to pages of the relevant classes. It might well be the case that this problem imposes greater difficulties upon a linear classifier such as RAINBOW, whereas the rule learner RIPPER is better able to focus on the small differences between similar pages in different classes. It might also be the case that the differences in the experimental setup (stemming,

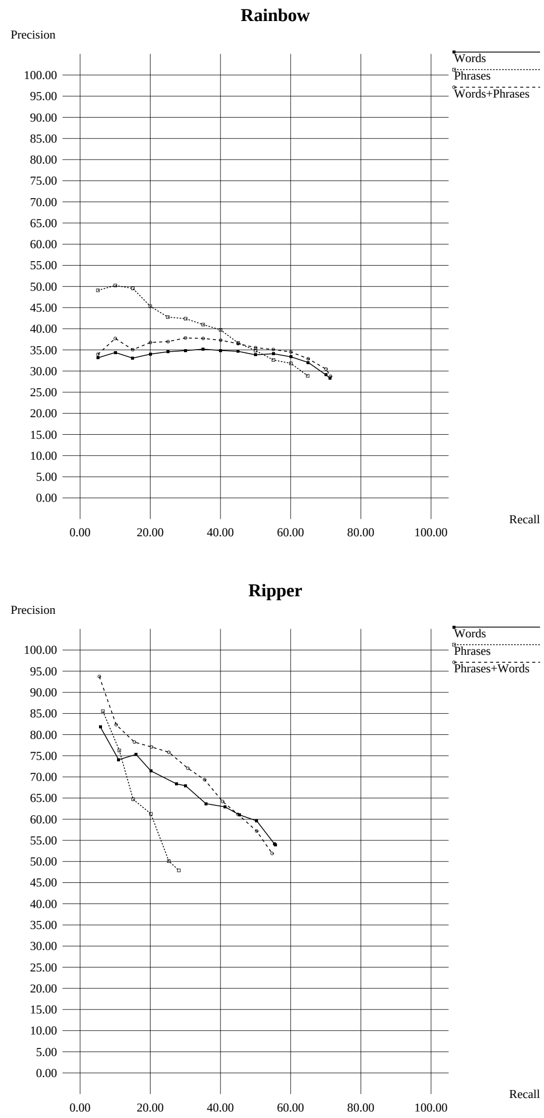

Figure 1: Combined precision/recall for RAINBOW (bottom) and RIPPER (top).

pruning of features that occur only once) contributed to this effect, but it cannot be its sole cause, because no stemming was used on the phrasal features in both cases.

Precision vs. Recall

More interesting, however, is a closer look at the precision/recall graphs shown in figure 1. In this graph, recall means the percentage of correctly predicted examples of the six relevant classes (not including *Other*), while precision means the percentage of correct predictions in all predictions for examples of these classes.

For generating these graphs, we associated a confidence score with each prediction. The confidence associated with a predictions of RAINBOW is simply the

class		phrase
student	5	I am <_>
	7	<_> is student
	14	student in <_>
faculty	12	university of <_>
	26	professor of <_>
	37	<_> is professor
staff	19	I am <_>
	26	<_> is associate
	30	manager of <_>
project	18	associated <_>
	28	related <_>
	58	affiliated <_>
department	12	department of <_>
	13	undergraduate <_>
	16	graduate <_>
course	25	<_> due
	34	due <_>
	40	fall <_>

Table 4: The best three phrases for each class and their rank in a sorted list of features in the words+phrases representation.

estimated probability of the example being in the predicted class. RIPPER does not provide probability estimates, so we associated with each prediction of RIPPER a confidence value $\frac{c+1}{c+i+2}$, where c (i) is the number of training examples that are correctly (incorrectly) classified by the rule that predicted the class for this example.[6] We chose the Laplace-estimate for estimating the accuracy of a rule in order to penalize rules that cover only a few examples.

We measured the precision of the algorithms at 5% increments in recall. A recall level of 5% is equivalent to correctly classifying about 55 of the 1091 examples of the six relevant classes. The precision at a recall level of $n\%$ was measured by first sorting the predictions for examples of the relevant classes according to their confidence score. Then we went down the list until the number of correct predictions exceeded $n\%$ of the total number of examples of the relevant classes, i.e., until we had recalled $n\%$ of the examples of these classes. The number of correct predictions over the total number of predictions in that segment of the list is the precision score associated with that recall level. If a recall of $n\%$ was reached within a series of predictions with identical confidence scores, we continued to process the list until the confidence score changed. Hence, in some cases, the points for the precision/recall curves are not exactly lined up at 5% increments.

```
     student :-  my,
                 student,
                 am,
                 DDD_DDDD.

Training Accuracy:          149 2
Test Accuracy (Washington):   9 3
```

```
     student :- I am <_>,
                <_> is student,
                institute of <_>.

Training Accuracy:           43 0
Test Accuracy (Texas):        5 0
```

```
     student :- my,
                student,
                <_> is student,
                DDD_DDDD.

Training Accuracy:          125 0
Test Accuracy (Texas):       12 0
```

Table 5: The rules with highest confidence scores for words, phrases, and phrases+words respectively, along with the number of correct and incorrect predictions they make on their respective training and test sets.

In both graphs, it is apparent that at lower recall levels, the phrasal features outperform the word-based representation. This supports the hypothesis that some phrasal features are highly predictive for certain classes, but in general have low coverage. This is particularly obvious in the significant decrease of the maximum recall for RIPPER if it uses only phrasal features.

The results for the combined representation are more diverse: RAINBOW assigns higher weights to word-based features than to phrasal features, so that the results for the combined representation are mostly determined by the words. Table 4 illustrates this by showing the top three phrases for each class, and their rank in the list of features ordered by a weighted log-odds ratio $\Pr(w_i|c)\log\left(\frac{\Pr(w_i|c)}{\Pr(w_i|\neg c)}\right)$. But even though the word-based features determine the shape of the curve, the addition of phrasal features results in small improvements at all recall levels.

The situation is quite similar for RIPPER in the sense that only a few of the learned rules actually use the phrasal features. However, the phrases frequently occur in rules with a high confidence score and make a crucial difference at the low recall end of the graph. Table 5 shows, for each of the three representations, the rule that was assigned the highest confidence score based on its performance on the respective training

[6]Note that these are not necessarily equivalent to the number of examples that are correctly or incorrectly classified by the rule in isolation, because RIPPER learns *decision lists*. This means that the learned rules are processed in order, and only examples that are not classified by previous rules can be classified by subsequent rules.

set. It is very interesting to observe that the best rule for the word-based feature set and the best rule for the words+phrases representation are almost identical. Both require the presence of the word *my* and a seven-digit number (most likely a phone number). However, while the word-based representation requires the presence of the words *am* and *student*, the feature-based representation requires the presence of the phrase <_> *is student*. Recall that all documents containing the sentence *"I am a student."* match both conditions, because the phrase matches all forms of the auxiliary verb *to be*. Looking at the accuracies of these two rules shows that the first one matches 163 examples in the entire data set, 5 of which being non-student pages, while the second rule matches only 137 examples, but all of them are of the class *Student*.

The rule that was formed using only the phrasal features can be loosely interpreted as classifying all documents that contain the sentence *"I am a student at the institute of <_>."* as student home pages. While this rule is sensible and accurate, it has a much lower coverage than both other rules.

It is also interesting to note that the third rule contains the redundant condition *student*. Apparently, RIPPER's information gain heuristic first preferred this condition over the more accurate phrase that contains the word, because the word feature had higher coverage. After it discovered that the phrase has to be added nevertheless, the redundant condition is not removed because RIPPER's pruning algorithm only considers the removal of a final sequence of conditions from a rule.[7]

It should be remarked that it is no coincidence that all three rules are predicting the class *Student*. In fact, most of the improvements at the low recall end of the curves is due to respective improvements in the prediction of the *Student* class. Precision/recall graphs for this class look very similar to the graphs shown in figure 1, while for the other five relevant classes no dramatic improvements could be observed. We have seen in table 4 that RAINBOW attributes a somewhat lower importance to phrasal features in the other 5 classes, and an investigation of the learned rules shows that only a few of the top-ranked rules for classes other than *Student* actually use phrasal features. This may be partly due to the fact that there are too few training examples for some of these classes. We plan to further investigate this on a more balanced data set, like the 20 newsgroups data used in (Lang 1995).[8]

Two other interesting observations to make in figure 1 are the differences in maximum recall between

[7]This is one of the differences between RIPPER and the original incremental reduced error pruning algorithm, which — less efficiently — considers all conditions as candidates for pruning (Fürnkranz & Widmer 1994).

[8]We have also investigated whether the *Student* class contains a higher percentage of natural language text, but we could not empirically confirm this hypothesis (using the crude measure *number of phrases per class* over *number of words per class*).

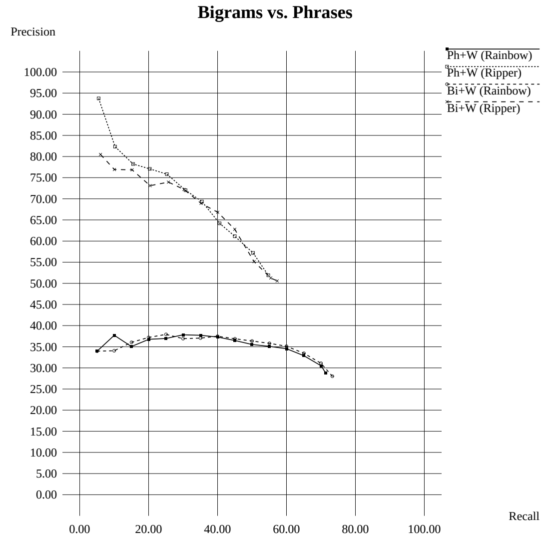

Figure 2: Precision/recall curves for RAINBOW (top) and RIPPER (bottom) using phrases+words versus bigrams+words.

RAINBOW and RIPPER, and the consistent decline of precision at the low recall end for RAINBOW. The former is due to fact that RAINBOW classifies less pages as *Other*, thus increasing the percentage of recalled pages in the relevant classes at the expense of precision. We conjecture that the latter phenomenon is caused by violations of the independence assumption of Naive Bayes, which leads to overly optimistic confidence scores for some predictions. For example, if the phrase <_> *is student* occurs in a document, it is quite likely that at least one of the phrases *student at/in/of* <_> will occur in the same sentence, which will unduly boost the probability that this document is a student home page.

Comparison to Bigrams

While the results of the last section have shown that phrasal features can improve the precision of text classifiers at the expense of recall, one can ask whether similar results could have been obtained by using sequences of words instead of single words as features. To this end, we have repeated some of the above experiments using a representation that considers single words and pairs of words (*bigrams*) as features.

For RAINBOW, we observed the same phenomenon as with the use of phrases: the shape of the recall/precision curve is determined by the word-based features, but the precision is slightly higher. The two curves at the bottom of figure 2 are the recall/precision curves for bigrams+words versus phrases+words for RAINBOW. There are no notable differences except a small peak for the phrase representation at a recall level of 10%. A comparison of the best bigram features (table 6) to the best phrase features (table 4) shows that the average rank of the top three features among the

class		bigram
student	4	`home page`
	7	`comput scienc`
	17	`depart of`
department	2	`comput scienc`
	4	`the depart`
	11	`scienc depart`
faculty	8	`comput scienc`
	10	`of comput`
	12	`univ of`
staff	4	`satoshi sekin`
	5	`rice edu`
	8	`in japanes`
project	12	`research group`
	16	`audio latex`
	17	`latex postscript`
course	9	`will be`
	14	`offic hour`
	19	`the cours`

Table 6: The best three bigrams for each class and their rank in a sorted list of features in the bigrams+words representation.

```
     student :- my,
                student,
                i_am,
                science,
                research.

Training Accuracy:          184 3
Test Accuracy (Washington):   9 3
```

Table 7: A highly ranked rule using bigrams with the number of correct and incorrect predictions it makes on training and test examples.

word-based features is higher for bigrams, while they appear to be less sensible (cf., e.g., the features for classes *Staff* and *Project*).

For RIPPER, however, the situation is different. In the upper two curves of figure 2, the phrase representation outperforms the bigram representation at the low recall end. Looking at the rules that apply there, we find that, unlike the rules of table 5, for the bigrams the rule with the highest confidence is not one of the 4 top-ranked rules of the *Student* class in the respective folds of the 4-fold cross-validation. The rule shown in table 7 is most similar to the rules shown in table 5. It is ranked number 6 by our confidence measure.

Finally, it is worth to take a look at the number of different features that are considered by the learning algorithms (table 8). Recall that we used RAINBOW with stemming on words and bigrams, as well as pruning of all features that occur only once, so the number of features it uses is much smaller than the number features RIPPER considers. It can be seen that, although there are slightly more different phrases than words, their numbers are in about the same order of magnitude, while the number of bigrams found in the documents is one order of magnitude larger.

	Words	Phrases	Bigrams
RAINBOW	26,628	36,216	224,751
RIPPER	92,666	116,990	872,275

Table 8: Number of features considered by RAINBOW and RIPPER

Conclusions

Our experiments have shown that the use of linguistic features can improve the precision of text categorization at the low recall end. For the rule learning algorithm RIPPER, adding such features was able to boost the precision by about 10% to more than 90% when recalling about 5% of the 1091 test examples of a text categorization task on the WWW. Although phrasal features require more engineering effort (e.g., the syntax of both the training and the test documents has to be parsed), they seemingly provide a better focus for rule learning algorithms. This effect was less remarkable for the naive Bayes classifier that we used.

Nevertheless, it should be noted that we were not able to improve the precision of the classifiers at high recall levels, the reason being that the phrasal features typically have a very narrow focus. However, it should also be noted that the test domain used in our case study may not have been an ideal choice for evaluating the utility of phrasal features, because significant parts of WWW pages do not contain natural language text.[9] Thus, we plan to further evaluate this technique on commonly used text categorization benchmarks, such as the 20 newsgroups dataset (Lang 1995) and the REUTERS newswire collection (Cohen & Singer 1996).

On the other hand, we are also working on further improving the classification accuracy on the 4 universities data set used in this case study. For example, all approaches used in this study performed very poorly on the *Project* class (precision was typically below 20%). The reason for this is most likely the heterogeneous nature of topics that are dealt with on project pages. We believe that this problem can be solved by looking at the information that is contained on or near the links that point to such pages and plan to investigate this topic further using techniques similar to those employed in this study.

[9]For example, one of the most characteristic words for the classes *Faculty* and *Staff* is the word "Fax", which is more likely to occur in addresses than in the natural language portion of a Web-page.

Acknowledgements

Johannes Fürnkranz is supported by a *Schrödinger-Stipendium* (J1443-INF) of the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung (FWF)*. Tom Mitchell acknowledges support by the Darpa HPKB program under contract F30602-97-1-0215. Ellen Riloff acknowledges support from NSF grants IRI-9509820 and IRI-9704240. We wish to thank Mark Schmelzenbach for generating the AutoSlog-TS data.

References

Cohen, W. W., and Singer, Y. 1996. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-96)*, 307–315.

Cohen, W. W. 1995. Fast effective rule induction. In Prieditis, A., and Russell, S., eds., *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, 115–123. Lake Tahoe, CA: Morgan Kaufmann.

Cohen, W. W. 1996. Learning trees and rules with set-valued features. In *Proceedings of the 13th National Conference on Artificial Intelligene (AAAI-96)*, 709–716. AAAI Press.

Craven, M.; DiPasquio, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 1998a. Learning to extract symbolic knowledge from the World Wide Web. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Craven, M.; DiPasquio, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 1998b. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*. AAAI Press.

Fürnkranz, J., and Widmer, G. 1994. Incremental Reduced Error Pruning. In Cohen, W., and Hirsh, H., eds., *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, 70–77. New Brunswick, NJ: Morgan Kaufmann.

Fürnkranz, J. 1997. Pruning algorithms for rule learning. *Machine Learning* 27(2):139–171.

Fürnkranz, J. 1998. Separate-and-conquer rule learning. *Artificial Intelligence Review*. In press.

Lang, K. 1995. NewsWeeder: Learning to filter netnews. In Prieditis, A., and Russell, S., eds., *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, 331–339. Morgan Kaufmann.

Mitchell, T. M. 1997. *Machine Learning*. McGraw Hill.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–266.

Riloff, E., and Lorenzen, J. 1998. Extraction-based text categorization: Generating domain-specific role relationships automatically. In Strzalkowski, T., ed., *Natural Language Information Retrieval*. Kluwer Academic Publishers. forthcoming.

Riloff, E. 1995. Little words can make a big difference for text classification. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 130–136.

Riloff, E. 1996a. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 1044–1049. AAAI Press.

Riloff, E. 1996b. An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence* 85:101–134.

Witten, I. H., and Bell, T. C. 1991. The zero-frequence problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory* 37(4):1085–1094.