A Study Using n-gram Features for Text Categorization

Johannes Fürnkranz

Austrian Research Institute for Artificial Intelligence

Schottengasse 3, A-1010 Wien, Austria

E-mail: juffi@ai.univie.ac.at

Technical Report OEFAI-TR-98-30

Abstract

In this paper, we study the effect of using n-grams (sequences of words of length n) for text categorization. We use an efficient algorithm for generating such n-gram features in two benchmark domains, the 20 newsgroups data set and 21,578 REUTERS newswire articles. Our results with the rule learning algorithm RIPPER indicate that, after the removal of stop words, word sequences of length 2 or 3 are most useful. Using longer sequences reduces classification performance.

1 Introduction

After Lewis' influential thesis (Lewis 1992c), the use of Machine Learning techniques for Text Categorization has gained in popularity (see, e.g., (Hearst and Hirsh 1996; Sahami 1998)). One requirement for the use of most Machine Learning algorithms is that the training data can be represented as a set of feature vectors. A straight-forward approach for representing text as feature vectors is the *set-of-words* approach: A document is represented by a feature vector that contains one boolean attribute for each word that occurs in the training collection of documents. If a word occurs in a particular training document, its corresponding attribute is set to 1, if not it is set to 0. Thus, each document is represented by the set of words it consists of.[1]

In this paper, we study the effect of generalizing the set-of-words approach by using word sequences, so-called n-grams, as features. We describe an algorithm for efficient generation and frequency-based pruning of n-gram features in section 2. In section 3 we present the results on two benchmark tasks, Ken Lang's 20 newsgroups data set and the 21,578 REUTERS newswire articles. The results indicate that word sequences of length 2 or 3 usually improve classification

[1] A related approach, the *bag-of-words* approach, uses the frequencies of occurrence of the individual words as feature values. The differences between both approaches in the context of naive Bayes classifiers were studied by McCallum and Nigam (1998).

performance, while longer sequences are not as useful. They also show that moderate frequency-based pruning of the feature set is useful, while heavy frequency-based pruning results in a performance decrease on the studied datasets.

2 Efficiently Generating n-gram Features

For small values of n, the number of different n-gram features that can be discovered in a collection of documents increases monotonically with n. For every n-gram there is at least one $n+1$-gram that has the n-gram as a starting sequence. The only exception is the final sequence in the document. n-grams that occur more than once will produce more than one $n+1$-gram if the different occurrences of the n-gram are followed by different words. On the other hand, for similar reasons, the number of occurrences of most n-grams will decrease with increasing n. Thus, although the number of features grows at least linearly with n, the number of features with a certain minimum frequency will grow much slower. An efficient algorithm for generating these feature sets should therefore avoid to generate all n-grams.

We implemented such an algorithm based on the APRIORI algorithm for efficiently generating association rules (Agrawal et al. 1995). The proposed technique is quite similar (if not identical) to the one that was (independently) developed by Mladenić and Grobelnik (1998). The basic idea of the algorithm is to utilize a user-specified lower bound on the minimum number of occurrences of a feature. n-grams that occur less frequently than this bound will not be used as features for the learning algorithm. For generating such pruned feature sets efficiently, the algorithm exploits a simple property:

Sub-sequence Property: *The number of occurrences of a sequence of m words in a document collection is bounded from above by the number of occurrences of each of its subsequences.*

This property can be exploited in order to obtain a simple but efficient algorithm. The n-gram features are generated by n different passes over the documents. In each pass, the number of occurrences of each feature is counted, and a user-specified threshold is used to prune infrequent features. In order to avoid the combinatorial explosion in the feature space, we can use the subsequence property for pruning the search space: We only have to count sequences of n words for which the sequences of the first $n-1$ and the last $n-1$ words have previously passed the frequency threshold. Other sequences can be ignored.

Figure 1 shows the resulting algorithm. It takes three parameters: the collection of *Documents*, the maximum length of the features (*MaxNGramSize*), and a lower bound on the number of occurrences of a feature (*MinFrequency*). The algorithm then computes all *Features* of length at most *MaxNGramSize* that occur at least *MinFrequency* times in the *Documents*.

For computing this result, it performs *MaxNGramSize* passes over the document collection, one for each possible feature length. In principle, however, one pass over the database would be sufficient. Instead of merely counting the occurrences of each word, the algorithm has to keep pointers to the positions where each feature in the text occurs. After computing this list of

```
procedure GENERATEFEATURES(Documents,MaxNGramSize,MinFrequency)

Features[0] = {∅}
for n = 1..MaxNGramSize
    Candidates = ∅
    Features[n] = ∅
    foreach Doc ∈ Documents
        foreach NGram ∈ NGrams(Doc,n)
            InitialGram = NGram − LastWord(NGram)
            FinalGram = NGram − FirstWord(NGram)
            if InitialGram ∈ Features[n − 1]
            and FinalGram ∈ Features[n − 1]
                Counter{NGram} = Counter{NGram} +1
                Candidates = Candidates ∪ NGram
    foreach NGram ∈ Candidates
        if Counter{NGram} ≥ MinFrequency
            Features[n] = Features[n] ∪ NGram
return Features
```

Figure 1: Efficiently generating features with an APRIORI-like algorithm.

pointers in the first pass over the documents, the feature set of length $n + 1$ can be computed from the feature set of length n by the following algorithm:

1. Find pairs of features that intersect (e.g. `find_pairs_of` and `pairs_of_features`)
2. For each such pair, compute the intersection of the position pointers of the two features. This is defined as the subset of the position pointers of the first feature for which a pointer to the immediately following position is contained in the set of position pointers of the second feature.
3. Discard all features for which the number of associated position pointers is below the frequency threshold.

This algorithm is inspired by the APRIORITID algorithm, which is also described in (Agrawal et al. 1995). It only has to read the documents once, but the memory requirements are much higher than for the algorithm of figure 1 because it has to store a list of position pointers for each feature (instead of using only a counter). For each iteration, the number of accesses to the hash table that stores these position pointers is quadratic in the number of features found in the previous iteration, while it is linear in the size of the document collection for the APRIORI-based algorithm. Consequently, we have found that additional passes over the document collection are cheaper if the number of features is large. Only for higher n-gram sizes, when the size of the feature sets becomes small (ca. ≤ 500), the use of position pointers begins to pay off.

We have implemented both algorithms in `perl`. The implementation has an additional parameter that can be used to specify with which iteration the mode should switch from making

additional passes through the document collection to using position indices. Another parameter allows the user to not only specify a *minimum term frequency* (number of times a feature occurs in the collection) but also a *minimum document frequency* (minimum number of documents in which a feature must appear). A feature will be accepted if it is above both thresholds.

3 Experimental Results

We used the inductive rule learning algorithm RIPPER for experiments in two domains: the 21578 REUTERS newswire data and Ken Lang's 20 newsgroups data set. In the following, we briefly describe RIPPER, our experimental setup, and the results in both domains.

3.1 RIPPER

William Cohen's RIPPER[2] (Cohen 1995) is an efficient, noise-tolerant rule learning algorithm based on the incremental reduced-error-pruning algorithm (Fürnkranz and Widmer 1994; Fürnkranz 1997). 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 boolean features, each encoding the presence or absence of a particular word (or n-gram) in that document. This results in a very inefficient encoding of the training examples because much space is wasted for specifying the 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, this does not differ from the use of boolean features in conventional learning algorithms, but RIPPER makes use of some clever optimizations. In the remainder of this paper, we will frequently continue to refer to each n-gram as a separate boolean feature.

3.2 Experimental Setup

For each of the two datasets, we represented each document with m set-valued features, one for each n-gram size $1 \leq m \leq$ *MaxNGramSize*. This means that all experiments using 3-grams also included 2-grams (*bigrams*) and 1-grams (*unigrams*). We generated several different versions of the datasets, for various settings of the parameters DF (*minimum document frequency*) and TF (*minimum term frequency*) as described at the end of section 2.

It is important to note that we used a stop-list[3] in order to reduce the number of n-grams. Many frequent n-grams that consist of a concatenation of frequent but uninformative prepositions and articles can be avoided that way. However, it should be mentioned that there is some evidence that important information might be thrown away with such a technique (see, e.g., (Riloff 1995)). We also ignored sentence boundaries, converted all characters to lower case, and replaced all digits with a 'D' and all special characters with an '_'.

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

[3] We used the stop list that is publicly available at `http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words`.

3.3 20 Newsgroups

The first dataset we experimented with was Ken Lang's 20-newsgroups data. This is a collection of 20,000 netnews articles, about 1,000 from each of 20 different newsgroups. The dataset is available from `http://www.cs.cmu.edu/afs/cs/project/theo-3/www/`. The task is to identify to which newsgroup an article belongs to.

We evaluated RIPPER with various feature sets using its built-in cross-validation. Because of the complexity, we chose to use only 5 folds. Note, however, that this procedure is problematic because of the characteristics of newsgroup articles: It happens quite frequently that portions of an articles are quoted in several subsequent articles of the same newsgroup. As such related articles may appear in both, training and test sets, there is a danger of over-optimistic accuracy estimates. However, we believe that the estimates are good enough for comparing different versions of the same learning setup.

Table 1 shows the results. The first column shows the pruning parameters. We measured the average error rate, the average run-time for the learning algorithm in CPU seconds (this does not include the time needed for generating the feature set), and the (cumulative) number of generated features for several different settings of the algorithm's parameters, and for several different maximal n-gram sizes. DF and TF stand for minimum document frequency and minimum term frequency, respectively. The *set-of-words* setting refers to the conventional text learning setting where each word is treated as a separate boolean feature.

The best results could be obtained with fairly moderate frequency-based pruning (all features that occur at least 5 times in at least 3 documents are admitted) and the use of sequences with maximum size 3. In all groups with identical pruning parameters (except for the ones with very heavy pruning), the use of n-grams improves the results. However, sequences of length > 3 do no longer improve the results (and make them worse in some cases). Frequency-based pruning works well if the parameter settings are fairly low, but the results get worse with increasing amounts of pruning. Obviously, several good features have a fairly low coverage and are thrown away with higher settings of the pruning parameters.

A look at the highest ranked features shows that they are not very indicative of any of the classes. The top ten features and their frequencies are shown in figure 2.

Obviously, none of the words are predictive of any of the classes. The first word that seems to be predictive for some classes (`soc.talk.religion.misc`, `soc.religion.christian`, and `alt.atheism`) is `god`, which is ranked 31 with 4550 occurrences. For higher n-gram sizes, the situation is similar. These problems could be alleviated by tailoring the stop list to the domain specifics. However, this not only requires a considerable effort but it also does not solve all problems: The repetitive nature of this domain (entire paragraphs may be repeated in several documents) may lead to overfitting. For example the fragment "closed roads mountain passes serve ways escape" produced the 4 highest ranked 4-grams that do not contain any numerical patterns or special characters, each one of them occurring 153 times. Most likely, an article that contains this passage has been quoted 152 times.

Pruning	n-grams	Error rate	CPU secs.	No. Features
set-of-words		47.07 ± 0.92	n.a.	71,731
	1	46.18 ± 0.94	12686.12	36,534
DF: 3	2	45.28 ± 0.51	15288.32	113,716
TF: 5	3	45.05 ± 1.22	15253.27	155,184
	4	45.18 ± 1.17	14951.17	189,933
	1	45.51 ± 0.83	12948.31	22,573
DF: 5	2	45.34 ± 0.68	13280.73	44,893
TF: 10	3	46.11 ± 0.73	12995.66	53,238
	4	46.11 ± 0.72	13063.68	59,455
	1	45.88 ± 0.89	10627.10	13,805
DF: 10	2	45.53 ± 0.86	13080.32	20,295
TF: 20	3	45.58 ± 0.87	11640.18	22,214
	4	45.74 ± 0.62	11505.92	23,565
	1	48.23 ± 0.69	10676.43	n.a.
DF: 25	2	48.97 ± 1.15	8870.05	n.a.
TF: 50	3	48.69 ± 1.04	10141.25	n.a.
	4	48.36 ± 1.01	10436.58	n.a.
	5	48.36 ± 1.01	10462.65	n.a.
	1	51.54 ± 0.60	8547.43	n.a.
DF: 50	2	49.71 ± 0.53	8164.27	n.a.
TF: 100	3	51.21 ± 1.26	8079.59	n.a.
	4	51.21 ± 1.26	8078.55	n.a.
	5	51.21 ± 1.26	8147.75	n.a.
	1	52.59 ± 0.71	6609.05	n.a.
DF: 75	2	52.83 ± 0.25	6532.80	n.a.
TF: 150	3	52.36 ± 0.48	6128.49	n.a.
	4	52.36 ± 0.48	6128.49	n.a.
	5	52.36 ± 0.48	6119.27	n.a.

Table 1: Results in the 20 newsgroups domain.

3.4 21578 REUTERS newswire data

The REUTERS newswire dataset has been frequently used as a benchmark for text categorization tasks. We used the version with 21,578 documents and evaluated it on the so-called ModApte-Split, which uses 9,603 documents for training and 3,299 for testing (and does not use the remaining documents). The standard evaluation procedure consists of a sequence of 90 binary classification tasks, one for each category. The results of these tasks are combined using micro-averaging. A more detailed description of this setup can be found in (Lewis 1997).

Figure 3 shows our results. We report recall and precision, the F1 value (the geometric mean between recall and precision), the predictive accuracy, and the number of features. In all

Feature	Frequency
ax	62063
D	61603
DD	48247
DDD	31188
_	19484
DDDD	18331
writes	14684
article	12567
dont	10255
like	10047

Table 2: 10 most frequent features in the 20 newsgroups domain.

representations it seems to be the case that the use of bigrams results in the highest recall and the lowest precision. In terms of F1 and predictive accuracy, bigrams have a clear advantage at moderate pruning, while with more heavy pruning, the unigrams representation seems to catch up.

It is also obvious that precision is correlated to the number of features. Unigrams give higher precision (but lower recall) than multi-grams, and an increase in the minimum frequency requirements also increases precision. For interpreting these results, it should be remembered that this domain is fairly simple, and for many of the classes the occurrence of a single word is sufficient to classify many of the articles.

A look at the features is not much different from the results in the 20 newsgroups domain: the most frequent features seem to bear no obvious relationship to any of the classes. Interesting is a comparison of the number of features: Although REUTERS contains only slightly more than 12,000 articles compared to the 20,000 of the 20 newsgroups dataset, the number of found features differs an order of magnitude. We think that the reasons for this phenomenon are that newsgroups articles are slightly longer on average, originate from a variety of authors and thus use a diverse vocabulary, the diversity of the topics of the newsgroups, and the repetitiveness of newsgroups articles which produces many n-grams by repetition of entire paragraphs of an article.

However, both, tables 1 and 3, exhibit a sub-linear growth of the number of features. Thus, the algorithm effectively avoids the super-linear growth of the number of features (see section 2).

4 Related Work

Feature generation and feature selection are important topics in information retrieval. Lewis (1992c) has emphasized their importance and studied several techniques on the REUTERS newswire data. Contrary to our results with n-gram features (in particular bigrams), Lewis (1992a) concludes that in the REUTERS dataset phrasal features (as well as term clustering)

Pruning	n-grams	Recall	Precision	F1	Accuracy	No. Features
set-of-words		76.71	83.42	79.92	99.5140	n.a.
	1	77.22	83.55	80.26	99.5211	9,673
DF: 3	2	80.34	82.03	81.18	99.5302	28,045
TF: 5	3	77.56	82.74	80.07	99.5130	38,646
	4	78.18	82.31	80.19	99.5130	45,876
	1	77.19	83.65	80.29	99.5221	6,332
DF: 5	2	80.05	82.06	81.04	99.5278	13,598
TF: 10	3	77.96	82.29	80.07	99.5106	17,708
	4	78.21	82.13	80.12	99.5106	20,468
	1	76.92	83.99	80.30	99.5241	4,068
DF: 10	2	79.06	82.04	80.52	99.5177	7,067
TF: 20	3	77.32	82.67	79.91	99.5096	8,759
	4	76.98	82.91	79.84	99.5096	9,907

Table 3: Results in the 21,578 REUTERS newswire domain.

provide no advantage over conventional set-of-words features. Notwithstanding these results, Fürnkranz, Mitchell, and Riloff (1998) could show that phrases can yield precision gains at low levels of recall.

Mladenić and Grobelnik (1998) performed a similar study using a naive Bayesian classifier for classifying WWW-documents into the hierarchy used by `www.yahoo.com`. They also conclude that sequences of length up to 3 can improve the performance, while longer sequences do not improve performance. The main difference to our study are the use of a different classifier, a different domain, and some differences in the setup of the experiments (e.g., Mladenić and Grobelnik (1998) used a fixed number of features, while we used a frequency-threshold for determining the number of features).

5 Discussion

We presented a simple but efficient algorithm for generating n-gram features and investigated their utility in two benchmark domains. The algorithm is based on the APRIORI-algorithm for discovering frequent item subsets in databases. A similar adaptation of the algorithm has been independently developed and studied by Mladenić and Grobelnik (1998). In both studies, the results seem to indicate that the addition of n-grams to the set-of-words representation frequently used by text categorization systems improves performance. However, sequences of length $n > 3$ are not useful and may decrease the performance.

Note that the results in this paper were obtained using a simple frequency-based feature subset selection. Although there is some evidence that frequency based pruning of feature sets is quite competitive in text categorization domains (Yang and Pedersen 1997; Mladenić 1998), it might be worth-while to study the use of more sophisticated pruning techniques that take the

class information into account. On the other hand, Yang and Pedersen (1997) and Lewis (1992b) report that heavy pruning may improve performance, which is not consistent with our results.

The main reason for our choice of frequency-based pruning was that it can be easily integrated into the APRIORI-based feature generation algorithm. In principle, however, any other feature subset selection technique could be used as a post-processor to the algorithm. Furthermore, some techniques could be directly integrated into the algorithm. The only condition that the algorithm imposes is that if a feature is acceptable to the pruning criterion, all its subsequences have to be acceptable as well. For some measures that do not implement this condition, upper and/or lower bounds on the measures could be implemented that allow to weed out unpromising candidates (such as, e.g., the techniques that are used for pruning candidate conditions with unpromising information gain bounds in C4.5 (Quinlan 1993) and FOIL (Quinlan 1990)). Extending the feature generation techniques used in this paper into that direction is subject to further research.

Acknowledgements

This work was performed during the author's stay at Carnegie Mellon University, which was funded by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number J1443-INF (*Schrödinger-Stipendium*).

References

AGRAWAL, R., H. MANNILA, R. SRIKANT, H. TOIVONEN, & A. I. VERKAMO (1995). Fast discovery of association rules. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI Press.

COHEN, W. W. (1995). Fast effective rule induction. In A. Prieditis and S. Russell (Eds.), *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, Lake Tahoe, CA, pp. 115–123. 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)*, pp. 709–716. AAAI Press.

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

FÜRNKRANZ, J., T. MITCHELL, & E. RILOFF (1998). A case study in using linguistic phrases for text categorization on the WWW. In M. Sahami (Ed.), *Learning for Text Categorization: Proceedings of the 1998 AAAI/ICML Workshop*, Madison, WI, pp. 5–12. AAAI Press. Technical Report WS-98-05.

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

HEARST, M. A. & H. HIRSH (Eds.) (1996). *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*. AAAI Press. Technical Report SS-96-05.

LEWIS, D. D. (1992a). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Devlopment in Information Retrieval*, pp. 37–50.

LEWIS, D. D. (1992b). Feature selection and feature extraction for text categorization. In *Proceedings of a Workshop on Speech and Natural Language*, Harriman, NY, pp. 212–217.

LEWIS, D. D. (1992c, February). *Representation and Learning in Information Retrieval*. Ph. D. thesis, University of Massachusetts, Department of Computer and Information Science, MA.

LEWIS, D. D. (1997, September). Reuters-21578 text categorization test collection. README file (V 1.2), available from `http://www.research.att.com/~lewis/reuters21578/README.txt`.

MCCALLUM, A. & K. NIGAM (1998). A comparison of event models for naive bayes text classification. In M. Sahami (Ed.), *Learning for Text Categorization: Proceedings of the 1998 AAAI/ICML Workshop*, Madison, WI, pp. 41–48. AAAI Press.

MLADENIĆ, D. (1998). Feature subset selection in text-learning. In *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*. Springer-Verlag.

MLADENIĆ, D. & M. GROBELNIK (1998). Word sequences as features in text learning. In *Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK-98)*, Ljubljana, Slovenia. IEEE section.

QUINLAN, J. R. (1990). Learning logical definitions from relations. *Machine Learning 5*, 239–266.

QUINLAN, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

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*, pp. 130–136.

SAHAMI, M. (Ed.) (1998). *Learning for Text Categorization: Proceedings of the 1998 AAAI/ICML Workshop*. AAAI Press. Technical Report WS-98-05.

YANG, Y. & J. O. PEDERSEN (1997). A comparative study on feature selection in text categorization. In D. Fisher (Ed.), *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pp. 412–420. Morgan Kaufmann.