Combining Unigrams and Bigrams in Semi-Supervised Text Classification

Igor Assis Braga, Maria Carolina Monard, Edson Takashi Matsubara

Mathematics and Computer Science Institute
University of Sao Paulo
Sao Carlos, SP 13566-590
Brazil
{igorab,mcmonard,edsontm}@icmc.usp.br

Abstract. Unlabeled documents vastly outnumber labeled documents in text classification. For this reason, semi-supervised learning is well suited to the task. Representing text as a combination of unigrams and bigrams has not shown consistent improvements compared to using unigrams in supervised text classification. Therefore, a natural question is whether this finding extends to semi-supervised learning, which provides a different way of combining multiple representations of data. In this paper, we investigate this question experimentally running two semi-supervised algorithms, Co-Training and Self-Training, on several text datasets. Our results do not indicate improvements by combining unigrams and bigrams in semi-supervised text classification. In addition, they suggest that this fact may stem from a strong "correlation" between unigrams and bigrams.

1 Introduction

Text Classification (TC) has received a great deal of attention from the machine learning community in the last ten years. The focus has primarily been on supervised classification, although there has also been interest in Semi-Supervised Learning (SSL). In SSL, one can derive a classifier from labeled and unlabeled data. This is particularly suited for many text classification tasks, since it is often the case that many unlabeled documents are available although labeling them is expensive.

A recurrent issue in TC is text representation. Extracting words, or *unigrams*, is by far the most common way to represent raw text in TC [1]. While this feature extraction technique is applicable to almost any kind of text, it has been found to be poor for certain classification tasks [2]. For dealing with this deficiency, word n-grams, *i.e.*, sequences of n consecutive words, were proposed by various researchers as a way to expand the standard unigram representation model [3].

Combinations of unigrams and n-grams, particularly *bigrams* ($n = 2$), have not shown consistent improvement in supervised text classification [1, 3]. Thus, a natural question arises: *does this behavior extend to semi-supervised learning?* In this paper, we investigate this question, which, to the best of our knowledge is yet unanswered. We conducted experiments on traditional text datasets using both the Co-Training and the Self-Training semi-supervised algorithms. The results do not indicate a major performance improvement by combining unigrams and bigrams in SSL. Based on these results and on other measurements taken during the execution of the Co-Training algorithm, we have found a possible explanation of why combining unigrams and bigrams does not help boost classification performance in SSL and even in the supervised case.

The remainder of this paper is organized as follows. In Section 2, we describe the Co-Training and the Self-Training algorithms. In Section 3, we review some related work and explain two different ways of combining unigrams and bigrams in semi-supervised learning. In Section 4, we present the experimental design and results. Finally, we discuss our findings and future work in Section 5.

2 Semi-Supervised Learning

Semi-Supervised Learning (SSL) has been applied to classification tasks in cases where labeled examples are scarce in comparison to available unlabeled examples. More precisely, we want to induce a classifier $h \rightarrow X \times Y$ using a set of j labeled examples $\{(\mathbf{x}_i, y_i)_{i=1}^{j}\}$ and a set of k unlabeled examples $\{(\mathbf{x}_i)_{i=j+1}^{j+k}\}$, with $k >> j$. Algorithms in SSL can be divided into *single-view* and *multi-view* algorithms. The latter requires that examples be represented by at least two different feature sets that are each nearly sufficient for classification on their own.

In the next section, we show two different ways of combining unigrams and bigrams in SSL. One of them uses a single-view algorithm, whereas the other requires a multi-view algorithm. In the remainder of this section, we present a single-view algorithm, Self-Training, and a multi-view one, Co-Training. Both of them were used in the experiments we describe later.

2.1 Self-Training

Self-Training is a wrapper semi-supervised algorithm [4] that makes use of a supervised learning algorithm that "trains" itself. Starting from a (possibly) small labeled set, the supervised learning algorithm derives a classifier, which is used to classify unlabeled examples that are later inserted into the labeled set. This expanded labeled set forms the basis of another induction step, and the process is repeated.

Algorithm 1 describes Self-Training. It takes as input a set L of labeled examples and a set U of unlabeled examples. Initially, a classifier h is derived using L and a supervised learning algorithm, which needs to produce classifiers that output a confidence score for its predictions. After h is derived, it is used to classify all examples in a randomly chosen $U' \in U$. The examples classified with the greatest confidence scores are selected by the function *bestExamples* and later inserted in L. If binary classification is all that matters, *bestExamples*

Algorithm 1: Self-Training

```
Input:
  – a set $L$ of labeled examples
  – a set $U$ of unlabeled examples

repeat
    use the examples in $L$ to derive a classifier $h$;
    randomly choose some examples from $U$, and insert them in $U'$;
    $L'$ = all examples in $U'$ labeled by $h$;
    $L'' = bestExamples(L')$;
    $L = L \cup L''$;
    remove from $U'$ all the examples in $L''$;
until $L'' = \emptyset$ ;
Output: $h$
```

can select the p most confident positive examples and the n most confident negative examples from L'. The algorithm stops when there are no more unlabeled examples available or other stopping criteria are met.

2.2 Co-Training

The Co-Training algorithm [5] is the multi-view counterpart of Self-Training. Interest in multi-view algorithms comes from the fact that they take advantage of multiple representations (views) of the data to produce better classifiers. Co-Training acts as if two Self-Training routines were executed simultaneously over the same sets L and U, the differences being that 1) multiple classifiers are derived, each one in a different view of the examples in L; and 2) the examples labeled by each classifier are also available to derive the other classifiers in the next iteration.

Hereafter, we assume that two views of the examples are available, so that an example $\mathbf{x}$ is a pair $(\mathbf{x}_1, \mathbf{x}_2)$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the values of the features that describes the example $\mathbf{x}$ in views 1 and 2 respectively. The general conditions that enable Co-Training to be successful are that $\mathbf{x}_1$ and $\mathbf{x}_2$ should not be too correlated and that the classifiers we want to learn in each view give the same label to $\mathbf{x}_1$ and $\mathbf{x}_2$ [6].

Algorithm 2 describes Co-Training. It is similar to Self-Training, although now the function *bestExamples* has to select the most confident examples classified by h_1 and h_2. In the original Co-Training algorithm [5], *bestExamples* selects the p most confident positive examples and the n most confident negative examples in each L'_1 and L'_2. With this version of *bestExamples* in mind, at most $2p + 2n$ examples are labeled at each iteration, since the same example can be selected from both L'_1 and L'_2.

Algorithm 2: Co-Training

```
Input:
  − a set $L$ of labeled examples
  − a set $U$ of unlabeled examples

repeat
    use view $\mathbf{x}_1$ of the examples in $L$ to derive a classifier $h_1$;
    use view $\mathbf{x}_2$ of the examples in $L$ to derive a classifier $h_2$;
    randomly choose some examples from $U$, and insert them in $U'$;
    $L'_1$ = all examples in $U'$ labeled by $h_1$;
    $L'_2$ = all examples in $U'$ labeled by $h_2$;
    $L'' = bestExamples(L'_1, L'_2)$;
    $L = L \cup L''$;
    remove from $U'$ all the examples in $L''$;
until $L'' = \emptyset$ ;
Output: $h_1, h_2$
```

3 Combining Unigrams and Bigrams in Text Classification

The study of unigram and bigram combinations started in supervised text classification. Next, we briefly discuss some previous work along this line. Following that, we explain two different ways of combining unigrams and bigrams in semi-supervised learning.

Supervised TC There are positive and negative results reported in the text classification literature on the subject of combining unigrams and bigrams. We review recent work, as good reviews can be found elsewhere [3, 7].

Bekkerman et al. [3] present a critical summary of both positive and negative results. They state that the majority of positive results are not significantly better than the baseline results of the datasets used, and, when they are, the baseline results tend to be very low. They also propose to combine unigrams and bigrams in supervised text classification using a feature generation method based on unigram and bigram clustering. However, the improvements achieved on the well known 20 Newsgroup dataset are not statistically significant.

Caropreso et al. [8] evaluate the usefulness of unigrams and bigrams in a learner-independent manner. They use some feature evaluation metrics to analyze the discrimination power of bigrams compared to that of unigrams. In the Reuters-21578 dataset, they find that there are bigrams that are more discriminative than unigrams. However, when more and more bigrams are selected at the expense of letting unigrams out of the feature set, the classification performance in the same dataset tends to degrade.

Pang et al. [2] analyze the use of machine learning techniques for sentiment classification in textual data. They represent text documents using, among others, unigrams, bigrams, and a single feature set that joins together unigrams and bigrams. This expanded feature set is then compared to what is achievable by using unigrams or bigrams alone on the Movie Review dataset. Their results show that the feature set composed of unigrams and bigrams does not reach a classification performance better than unigrams.

Semi-supervised TC As far as we know, there was no attempt to study unigram and bigram combinations in semi-supervised text classification. Previous work in semi-supervised TC has only considered the use of unigrams for text representation [9–11]. Therefore, it would be interesting to verify if this combination can improve classification performance in semi-supervised TC.

As a first approach, we can combine unigrams and bigrams as it has been done in supervised learning: joining the two representations together and using a single-view semi-supervised algorithm to learn a classifier from it. However, it should be observed that the way unigrams and bigrams will interact with each other is fully determined by the learning algorithm used.

As we described in the last section, a multi-view semi-supervised algorithm can be used when examples can be represented by at least two different feature sets that are each nearly sufficient for classification on their own. As our experiments in the next section suggest, this happens to be the case of unigrams and bigrams. An interesting characteristic of this combination is that each representation is kept in its own "world", since two classifiers are independently induced in each representation.

4 Experimental Evaluation

We conducted experiments to evaluate unigram and bigram combinations in semi-supervised learning. Particularly, we tried a single-view combination using Self-Training and a multi-view combination using Co-Training. We also ran Self-Training using unigrams and bigrams alone for comparison.

4.1 Datasets

Five text datasets were used in the experiments. One of them consists of the web pages view from the Courses dataset [5]; three are subsets of the UseNet news articles 20 Newsgroups dataset[1]; and the last one is the `polarity_dataset v2.0` from the Movie Review Data[2] — Table 1. All datasets except Courses are balanced.

The datasets were decomposed into the attribute-value format, where unigrams (1-gram) were used as one view and bigrams (2-gram) as the second view of the datasets. To this end, we used PreTexT II[3], a locally-developed text pre-processing tool. Stop-word removal and stemming [12] were carried out for each

[1] `http://people.csail.mit.edu/jrennie/20Newsgroups/`
[2] `http://www.cs.cornell.edu/People/pabo/movie-review-data/`
[3] `http://www.icmc.usp.br/~caneca/pretext.htm`

Table 1. Datasets description and attributes

Dataset	#Doc	View	#Attr	#Attr. P	Class	%Class
Courses	1050	1-gram	12254	3313	course	22%
					non-course	78%
		2-gram	46421	2073	course	22%
					non-course	78%
Hardware	1943	1-gram	13398	3958	pc	50%
					mac	50%
		2-gram	47331	2846	pc	50%
					mac	50%
Vehicles	1984	1-gram	14048	5362	car	50%
					motorcycles	50%
		2-gram	51404	3605	car	50%
					motorcycles	50%
Sports	1993	1-gram	14254	5741	baseball	50%
					hockey	50%
		2-gram	60114	4548	baseball	50%
					hockey	50%
Movie	2000	1-gram	25302	10669	pos. review	50%
					neg. review	50%
		2-gram	299423	9186	pos. review	50%
					neg. review	50%

dataset. After that, unigrams and bigrams that appeared in 3 or less documents were removed. The result of this pre-processing step is tabulated in Table 1, which shows the dataset name (Dataset); number of documents in the dataset (#Doc); number of generated attributes (#Attr); number of attributes left after pre-processing (#Attr. P); and class distribution (%Class).

4.2 Methodology

The supervised learning algorithm we used in Self-Training and Co-Training was the *Multinomial Naive Bayes* (MNB) [13]. In order to obtain a lower bound of the error that the algorithms can reach in these datasets, we measured the error rate of MNB using the full datasets and 10-fold cross-validation. Results (AUC and mean error rate) are shown in Table 2. It can be observed that, although there are no significant differences related to the AUC values for all the datasets, the error rate (%Error) is a little higher using bigrams (2-gram). It is also possible to observe that except for the Movie dataset and for the Hardware dataset using bigrams, the error rate of the classifiers is low. Moreover, using unigrams and bigrams together as a single feature set (1+2-gram) does not always improve classification performance compared to unigrams (1-gram), and, when it does, it is by a small margin. These results are in accordance with our analysis of previous work.

As the datasets we use contain only labeled examples, we ran Self-Training and Co-Training in a simulation mode, in which the true labels of an expressive

Table 2. *Multinomial Naive Bayes* results

Dataset	View	*Multinomial Naive Bayes* AUC	*Multinomial Naive Bayes* %Error
Courses	1-gram	0.97 (0.01)	4.57 (1.61)
	2-gram	0.98 (0.01)	5.24 (1.03)
	1+2-gram	0.97 (0.01)	4.86 (1.45)
Hardware	1-gram	0.98 (0.01)	6.23 (1.37)
	2-gram	0.96 (0.01)	12.15 (2.23)
	1+2-gram	0.98 (0.01)	6.38 (1.69)
Vehicles	1-gram	0.99 (0.01)	2.27 (1.33)
	2-gram	0.99 (0.01)	4.84 (2.13)
	1+2-gram	0.99 (0.01)	1.82 (1.19)
Sports	1-gram	0.99 (0.03)	1.15 (0.79)
	2-gram	0.99 (0.01)	3.01 (0.67)
	1+2-gram	0.99 (0.01)	0.95 (0.65)
Movie	1-gram	0.87 (0.03)	19.00 (2.99)
	2-gram	0.84 (0.04)	23.55 (3.32)
	1+2-gram	0.87 (0.03)	18.60 (3.25)

number of examples are hidden from the algorithms to create the unlabeled set. Furthermore, to assess the behavior of both algorithms using 10 cross-validation, the sampling method was adapted as follows: first, the examples in both views are paired and marked with an ID. Then, the folds are sampled so that both training and test samples are compatible, *i.e.*, an example identified with a given ID appears only in the training or in the test sample in both views.

All experiments were carried out using the same number of initial labeled examples L, distributed accordingly to the class distribution of the complete dataset. As in SSL we are concerned with small labeled sets, we fixed this number as 30, which corresponds to 1.5% of the number of examples in the largest dataset. It is possible to use the class distribution of the complete dataset because we are executing the algorithms in a simulation mode, where the true labels of the datasets are known. Previous experiments have shown that the best results are obtained if the true class distribution of the examples is known [14]. However, in a real case, it would be unwise to estimate this distribution using the few labeled examples in L.

For the sake of simplicity, consider that examples belong to two possible classes $\{\oplus, \ominus\}$. In each iteration of Self-Training, p examples most confidently classified as $\oplus$ and n examples most confidently classified as $\ominus$ are selected. The confidence score is measured by the MNB estimative of $P(\oplus|\mathbf{x})$. As for Co-Training, the same procedure is carried out for each view, and the confidence score is measured by the estimative of $P(\oplus|\mathbf{x}_1)$ in the first view and $P(\oplus|\mathbf{x}_2)$ in the second view. Moreover, if an example is selected in both views and $h_1(\mathbf{x}_1) \neq h_2(\mathbf{x}_2)$, the chosen label is the one given with higher confidence (ties broken randomly). For the unbalanced dataset Courses $(p, n) = (2, 8)$, and for the remaining datasets $(p, n) = (5, 5)$. Considering the different class distribution of the datasets, $p+n = 10$ is the minimum number that covers the class distribution of all the datasets. This means that in each iteration, 10 examples from U' are labeled by Self-Training while a minimum of 10 and a maximum of 20 examples can be labeled by Co-Training.

4.3 Results

In what follows, 1-gram represents the unigram view representation, 2-gram the bigram representation, and 1+2-gram represents the two views joined together in Self-Training. Tables 3 and 4 summarize the experimental results. We use the AUC, mean error rate (%Error) and the number of incorrectly labeled examples (#Errors) to assess the experiments. Values between brackets refer to the standard deviation. The AUC and the mean error rate refer to the classification on the test set, *i.e.* to the final classifiers generated by Self-Training or Co-Training. On the other hand, #Errors refers to the mean number of incorrectly labeled examples during the training phase, where the set L is incremented with new labeled examples.

Table 3. Self-Training results

		Self-Training		
Dataset	View	AUC	%Error	#Errors
Courses	1-gram	0.96 (0.01)	5.43 (1.68)	45.20 (3.80)
	2-gram	0.95 (0.02)	6.29 (1.50)	62.60 (8.11)
	1+2-gram	0.96 (0.01)	5.24 (1.97)	40.40 (2.07)
Hardware	1-gram	0.81 (0.12)	24.61 (10.80)	424.00 (130.93)
	2-gram	0.76 (0.15)	28.94 (10.51)	474.25 (198.17)
	1+2-gram	0.86 (0.06)	18.76 (6.27)	341.89 (105.52)
Vehicles	1-gram	0.99 (0.01)	3.88 (1.57)	74.50 (15.96)
	2-gram	0.98 (0.01)	5.44 (1.03)	137.60 (31.96)
	1+2-gram	0.99 (0.01)	3.28 (1.35)	62.50 (4.88)
Sports	1-gram	0.99 (0.01)	2.16 (0.98)	39.40 (9.74)
	2-gram	0.99 (0.01)	4.07 (0.87)	76.20 (22.72)
	1+2-gram	0.99 (0.01)	1.91 (0.82)	36.90 (8.29)
Movie	1-gram	0.66 (0.07)	36.30 (5.86)	650.40 (85.46)
	2-gram	0.60 (0.05)	42.90 (3.84)	759.40 (65.32)
	1+2-gram	0.65 (0.07)	37.45 (6.03)	643.80 (123.36)

As can be observed in Table 3, Self-Training results do not show a consistent improvement by joining together unigrams and bigrams (1+2-gram). Although the worst results are the ones where only bigrams (2-gram) are used, results using only unigrams (1-gram) are compatible with the ones obtained using 1+2-gram, except for the Hardware dataset.

In Co-Training, if we do not allow the classifiers h_1 and h_2 to label examples for each other, we are left with Self-Training using unigrams and bigrams

Table 4. Co-Training results

Dataset	View	Co-Training AUC	Co-Training %Error	Co-Training #Errors
Courses	1-gram	0.97 (0.01)	5.52 (2.19)	
	2-gram	0.96 (0.02)	6.00 (1.49)	44.50 (1.96)
Hardware	1-gram	0.87 (0.06)	20.28 (7.76)	
	2-gram	0.84 (0.07)	23.88 (6.96)	378.33 (39.80)
Vehicles	1-gram	0.99 (0.01)	4.36 (1.45)	
	2-gram	0.98 (0.01)	6.38 (0.30)	102.30 (27.65)
Sports	1-gram	0.99 (0.01)	1.96 (1.10)	
	2-gram	0.99 (0.01)	3.96 (1.19)	42.80 (15.20)
Movie	1-gram	0.67 (0.09)	37.55 (7.42)	
	2-gram	0.65 (0.09)	38.70 (7.36)	681.10 (128.16)

alone (remember from Section 2 that Co-Training is the multi-view counterpart of Self-Training). For this reason, combining unigrams and bigrams with Co-Training is worthwhile only when the classifiers obtained by Co-Training achieve better classification performance than the classifiers obtained by Self-Training using unigrams or bigrams alone. Observing Tables 3 and 4, it is possible to verify that Co-Training is not better than Self-Training using unigrams (1-gram) or bigrams (2-gram) on the Vehicles and Movie datasets. On the Hardware dataset, the inverse situation occurrs, and on the other datasets, the results of both are similar.

5 Discussion and Future Work

The results reported in the previous section do not show an improvement in classification performance when we combine unigrams and bigrams in a single-view and in a multi-view semi-supervised setting. Pursuing a possible explanation for these results, we plotted, for each dataset, a histogram of the mean number of examples correctly and incorrectly labeled by Co-Training versus $p_1 + p_2$, where $p_1 = P(\oplus|\mathbf{x}_1)$ and $p_2 = P(\oplus|\mathbf{x}_2)$. The value of $p_1 + p_2$ for a given example $\mathbf{x}$ will be close to 0 if both h_1 and h_2 classify $\mathbf{x}$ as $\ominus$ with high confidence. On the other hand, the value of $p_1 + p_2$ will be close to 2 if both h_1 and h_2 classify $\mathbf{x}$ as $\oplus$ with high confidence. Figures 1, 2, and 3 show some histograms for the Sports, Hardware, and Movie datasets. All other histograms can be found in `http://www.icmc.usp.br/~igorab/hist/`.

It is possible to observe in Figure 1 that most of the correctly labeled examples in the Sports dataset fall in the bins where $p_1 + p_2$ equals or is next to 0 or 2. In other words, for most of the examples labeled by Co-Training in this dataset, the classifiers h_1 and h_2 agree in their classification with high confidence. Simply put, this means that one classifier did not help the other, what causes the results in this dataset to be very close to the ones of Self-Training using unigrams or bigrams alone.

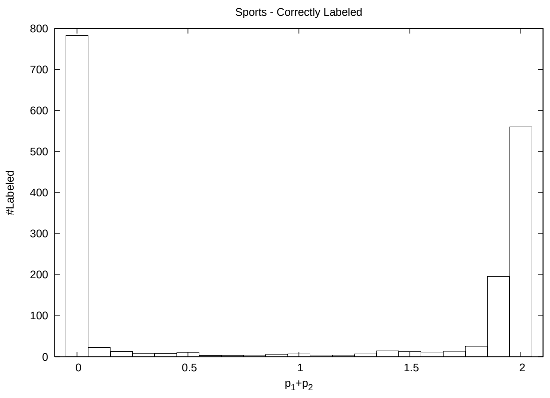

Fig. 1. Number of correctly labeled examples (*#Labeled*) versus $p_1 + p_2$ in the SPORTS dataset

The same pattern was observed in the histograms of the other datasets. In the HARDWARE dataset, for which the results are favorable to combining unigrams and bigrams, most of the correctly labeled examples still fall near 0 or 2, although the histogram in Figure 2 shows relatively more correctly labeled examples in other bins. Furthermore, in the MOVIE dataset, for which both CO-TRAINING and SELF-TRAINING perform equally bad, the histogram in Figure 3 shows that most of the incorrectly labeled examples are next to bins 0 and 2. In other words, both classifiers h_1 and h_2 agree with high confidence in incorrectly labeling these examples.

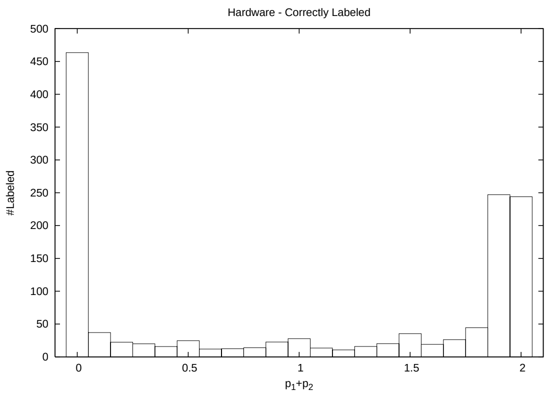

Fig. 2. Number of correctly labeled examples (*#Labeled*) versus $p_1 + p_2$ in the HARDWARE dataset

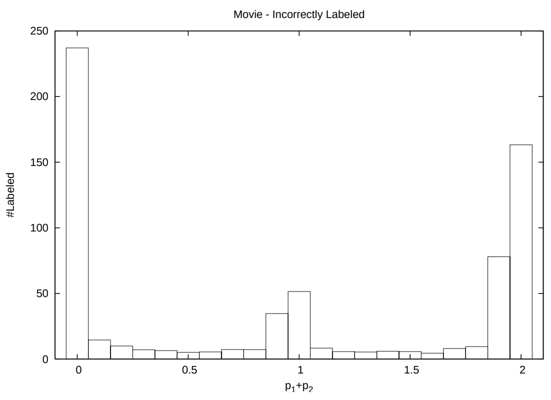

Fig. 3. Number of incorrectly labeled examples (*#Labeled*) versus p_1+p_2 in the MOVIE dataset

In summary, these histograms show a case of strong *view correlation* for unigrams and bigrams, since the classifiers derived on each of them predicts equally with high confidence in most of the examples. However, one of the requirements for CO-TRAINING to succeed is that the views should not be too correlated [6]. This seems a plausible explanation for why combining unigrams and bigrams in CO-TRAINING does not give better results than SELF-TRAINING using unigrams and bigrams alone. We conjecture that this view correlation is also responsible for the small improvements observed when unigrams and bigrams are joined together in SELF-TRAINING and in plain supervised learning.

As future work, we plan to extend the experimental evaluation to include more datasets. We also want to explore a text representation scheme based on carefully selected bigrams (or even trigrams). We would like to combine unigrams only to n-grams that, in a given domain, are more meaningful together than apart. This selection of bigrams can start by using methods for term extraction, which is a subject widely studied in the Natural Languague Processing community [15, 16].

Acknowledgments. The authors wish to thank research fund agencies FAPESP and CNPq for their continuous support.

References

1. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34**(1) (2002) 1–47
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: EMNLP '02: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing. (2002) 79–86

3. Bekkerman, R., Allan, J.: Using bigrams in text categorization. Technical report, University of Massachusetts (2003) `www.cs.umass.edu/~ronb/papers/bigrams.pdf`.
4. Chapelle, O., Schölkopf, B., Zien, A.: Introduction to semi-supervised learning. In: Semi-Supervised Learning (Adaptive Computation and Machine Learning). (2006) 1–12
5. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with Co-Training. In: COLT '98: Proceedings of the 11th Annual Conference on Computational Learning Theory. (1998) 92–100
6. Balcan, M.F., Blum, A., Yang, K.: Co-Training and expansion: Towards bridging theory and practice. In: NIPS '04: Advances in Neural Information Processing Systems 17. (2005) 89–96
7. Tan, C.M., Wang, Y.F., Lee, C.D.: The use of bigrams to enhance text categorization. Information Processing and Management **38**(4) (2002) 529–546
8. Caropreso, M.F., Matwin, S., Sebastiani, F.: A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In: Text databases & document management: theory & practice. (2001) 78–102
9. Nigam, K., McCallum, A., Mitchell, T.: Semi-supervised text classification using em. In: Semi-Supervised Learning (Adaptive Computation and Machine Learning). (2006) 33–53
10. Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of Co-Training. In: CIKM '00: Proceedings of the 9th International Conference on Information and Knowledge Management. (2000) 86–93
11. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML '99: Proceedings of the 16th International Conference on Machine Learning. (1999) 200–209
12. Porter, M.F.: An algorithm for suffix stripping. Program: electronic library and information systems **40**(3) (2006) 211–218
13. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. Technical Report WS-98-05, Association for the Advancement of Artificial Intelligence (1998) `http://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-007.pdf`.
14. Matsubara, E.T., Monard, M.C., Prati, R.C.: On the class distribution labelling step sensitivity of Co-Training. In: IFIP AI '06: Artificial Intelligence in Theory and Practice. (2006) 199–208
15. Dias, G., Guillore, S., Bassano, J.C., Lopes, J.G.P.: Combining linguistics with statistics for multiword term extraction: A fruitful association? (2000) 1473–1491
16. Pantel, P., Lin, D.: A statistical corpus-based term extractor. In: AI '01: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence. (2001) 36–46