Boosting automatic lexical acquisition with morphological information[*]

Massimiliano Ciaramita
Department of Cognitive and Linguistic Sciences
Brown University
Providence, RI, USA 02912
massimiliano_ciaramita@brown.edu

Abstract

In this paper we investigate the impact of morphological features on the task of automatically extending a dictionary. We approach the problem as a pattern classification task and compare the performance of several models in classifying nouns that are unknown to a broad coverage dictionary. We used a boosting classifier to compare the performance of models that use different sets of features. We show how adding simple morphological features to a model greatly improves the classification performance.

1 Introduction

The incompleteness of the available lexical resources is a major bottleneck in natural language processing (NLP). The development of methods for the automatic extension of these resources might affect many NLP tasks. Further, from a more general computational perspective, modeling lexical meaning is a necessary step toward semantic modeling of larger linguistic units.

We approach the problem of lexical acquisition as a classification task. The goal of the classifier is to insert new words into an existing dictionary. A dictionary[1] in this context simply associates lexical forms with class labels; e.g., $dog \rightarrow ANIMAL$, where the arrow can be interpreted as the ISA relation. In this study we use a simplified version of Wordnet as our base lexicon and we ignore other relevant semantic relations (like hyponymy) and the problem of word sense ambiguity. We focus on finding features that are useful for associating unknown words with class labels from the dictionary. In this paper we report the following preliminary findings. First of all we found that the task is difficult. We developed several models, based on nearest neighbor (NN), naive Bayes (NB) and boosting classifiers. Unfortunately, the error rate of these models is much higher than what is found in text categorization tasks[2] with comparable numbers of classes. Secondly, it seems obvious that information that is potentially useful for word classification can be of very diverse types, e.g., semantic and syntactic, morphological and topical. Therefore methods that allow flexible feature combination and selection are desirable. We experimented with a multiclass boosting algorithm (Schapire and Singer, 2000), which proved successful in this respect. In this context boosting combines two sources of information: words co-occurring near the new word, which we refer to as *collocations*, and morphological properties of the new word. This classifier shows improved performance over models that use only collocations. In particular, we found that even rudimentary morphological information greatly im-

[*] I would like to thank for their input everybody in the Brown Laboratory for Linguistic Information Processing (BLLIP) and Information Retrieval and Machine Learning Group at Brown (IRML), and particularly Mark Johnson and Thomas Hofmann. I also thank Brian Roark and Jesse Hochstadt.

[1] Or lexicon, we use the two terms interchangeably.

[2] Text categorization is the task of associating documents with topic labels (*POLITICS*, *SPORT*, ...) and it bears similarities with semantic classification tasks such as word sense disambiguation, information extraction and acquisition.

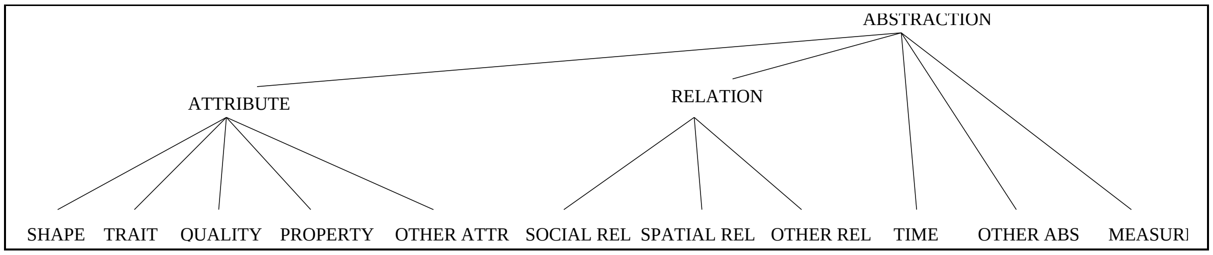

Figure 1: A few classes under the root class *ABSTRACTION* in MiniWordnet.

proves classification performance and should therefore be part of any word classification model.

The outline of the paper is as follows. In section 2 we introduce the dictionary we used for our tests, a simplified version of Wordnet. In section 3 we describe more formally the task, a few simple models, and the test methods. In section 4 we describe the boosting model and the set of morphological features. In section 5 we summarize the results of our experiments. In section 6 we describe related work, and then in section 7 we present our conclusions.

2 MiniWordnet

Ideally the lexicon we would like to extend is a broad coverage machine readable dictionary like Wordnet (Miller et al., 1990; Fellbaum, 1998). The problem with trying to directly use Wordnet is that it contains too many classes (synsets), around 70 thousand. Learning in such a huge class space can be extremely problematic, and intuitively it is not the best way to start on a task that hasn't been much explored[3]. Instead, we manually developed a smaller lexicon dubbed *MiniWordnet*, which is derived from Wordnet version 1.6. The reduced lexicon has the same coverage (about 95 thousand noun types) but only a fraction of the classes. In this paper we considered only nouns and the noun database. The goal was to reduce the number of classes to about one hundred[4] of roughly comparable taxonomical generality and consistency, while maintaining a little bit of hierarchical structure.

[3]Preliminary experiments confirmed this; classification is computationally expensive, performance is low, and it is very hard to obtain even very small improvements when the full database is used.

[4]A magnitude comparable to the class space of well studied text categorization data sets like the Reuters-21578 (Yang, 1999).

The output of the manual coding is a set of 106 classes that are the result of merging hundreds of synsets. A few random examples of these classes are *PERSON, PLANT, FLUID, LOCATION, ACTION*, and *BUSINESS*. One way to look at this set of classes is from the perspective of named-entity recognition tasks, where there are a few classes of a similar level of generality, e.g, *PERSON, LOCATION, ORGANIZATION, OTHER*. The difference here is that the classes are intended to capture all possible taxonomic distinctions collapsed into the *OTHER* class above. In addition to the 106 leaves we also kept a set of superordinate levels. We maintained the 9 root classes in Wordnet plus 18 intermediate ones. Examples of these intermediate classes are *ANIMAL, NATURAL_OBJECT, ARTIFACT, PROCESS*, and *ORGANIZATION*. The reason for keeping some of the superordinate structure is that hierarchical information might be important in word classification; this is something we will investigate in the future. For example, there might not be enough information to classify the noun *ostrich* in the *BIRD* class but enough to label it as *ANIMAL*. The superordinates are the original Wordnet synsets. The database has a maximum depth of 5.

We acknowledge that the methodology and results of reducing Wordnet in this way are highly subjective and noisy. However, we also think that going through an intermediary step with the reduced database has been useful for our purposes and it might also be so for other researchers[5]. Figure 1 depicts the hierarchy below the root class *ABSTRACTION*. The classes that are lined up at the bottom of the figure are leaves. As in Wordnet, some sub-

[5]More information about MiniWordnet and the database itself are available at `www.cog.brown.edu/∼massi/research`.

hierarchies are more densely populated than others. For example, the *ABSTRACTION* sub-hierarchy is more populated (11 leaves) than that of *EVENT* (3 leaves). The most populated and structured class is *ENTITY*, with almost half of the leaves (45) and several superordinate classes (10).

3 Automatic lexical acquisition

3.1 Word classification

We frame the task of inserting new words into the dictionary as a classification problem: $\mathcal{Y}$ is the set of classes defined by the dictionary. Given a vector of features $\vec{x} \in \mathcal{X} \subset \Re$ we want to find functions of the form $\mathcal{X} \rightarrow \mathcal{Y}$. In particular we are interested in learning functions from data, i.e., a training set of pairs $(\vec{x}, y)$, $\vec{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$, such that there will be a small probability of error when we apply the classifier to unknown pairs (new nouns).

Each class is described by a vector of features. A class of features that intuitively carry semantic information are *collocations*, i.e., words that co-occur with the nouns of interest in a corpus. Collocations have been widely used for tasks such as word sense disambiguation (WSD) (Yarowsky, 1995), information extraction (IE) (Riloff, 1996), and named-entity recognition (Collins and Singer, 1999). The choice of collocations can be conditioned in many ways: according to syntactic relations with the target word, syntactic category, distance from the target, and so on.

We use a very simple set of collocations: each word w that appears within $\pm k$ positions from a noun n is a feature. Each occurrence, or token, i of n, n_i, is then characterized by a vector of feature counts $\vec{n_i}$. The vector representation of the noun type n is the sum of all the vectors representing the contexts in which it occurs. Overall the vector representation for each class in the dictionary is the sum of the vectors of all nouns that are members of the class

$$\vec{y} = \sum_{n \in y} \sum_i \vec{n_i}$$

while the vector representation of an unknown noun is the sum of the feature vectors of the contexts in which it occurred

$$\vec{x} = \sum_i \vec{n_i}$$

The corpus that we used to collect the statistics about collocations is the set of articles from the 1989 Wall Street Journal (about 4 million words) in the BLLIP'99 corpus.

We performed the following tokenization steps. We used the Wordnet "morph" functions to morphologically simplify nouns, verbs and adjectives. We excluded only punctuation; we did no filtering for part of speech (POS). Each word was actually a word-POS pair; i.e., we distinguished between *plant*:NN and *plant*:VB. We collapsed sequences of NNs that appeared in Wordnet as one noun; so we have one entry for the noun *car_company*:NN. We also collapsed sequences of NNPs, possibly interleaved by the symbol "&", e.g., *George_Bush:NNP* and *Procter_&_Gamble*:NNP. To reduce the number of features a little we changed all NNPs beginning with Mr. or Ms. to *MISS_X*:NNP, all NNPs ending in *CORP.* or *CO.* to *COMPANY_X*:NNP, and all words with POS CD, i.e., numbers, starting with a digit to *NUMBER_X*:CD. For training and testing we considered only nouns that are not ambiguous according to the dictionary, and we used only features that occurred at least 10 times in the corpus.

3.2 Simple models

We developed several simple classifiers. In particular we focused on nearest neighbor (NN) and naive Bayes (NB) methods. Both are very simple and powerful classification techniques. For *NN* we used cosine as a measure of distance between two vectors, and the classifier is thus

$$g(\vec{x}) = \text{argmax}_{\vec{y}} \ \cos(\vec{x}, \vec{y}) \qquad (1)$$

Since we used aggregate vectors for classes and noun types, we only used the best class; i.e., we always used 1-nearest-neighbor classifiers. Thus k in this paper refers only to the size of the window around the target noun and never to number of neighbors consulted in k-nearest-neighbor classification. We found that using TFIDF weights instead of simple counts greatly improved performance of the *NN* classifiers, and we mainly report results relative to the TFIDF *NN* classifiers (NN_{TFIDF}). A document in this context is the context, delimited by the window size k, in which each each noun occurs. TFIDF basically filters out the impact of closed class

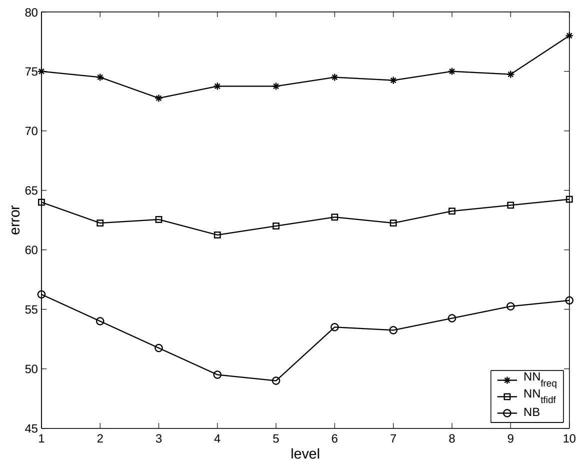

Figure 2: Error of the NN_{freq}, NN_{TFIDF} and NB models for $k = 1..10$.at level 1

words and re-weights features by their informativeness, thus making a stop list or other feature manipulations unnecessary. The naive Bayes classifiers is also very simple

$$f(\vec{x}) = \text{argmax}_y P(y) \prod_i P(x_i|y)) \qquad (2)$$

The parameters of the prior and class-conditional distributions are easily estimated using maximum likelihood. We smoothed all counts by a factor of .5.

3.3 Testing procedure

We tested each model on an increasing numbers of classes or *level*. At level 1 the dictionary maps nouns only to the nine Wordnet roots; i.e., there is a very coarse distinction among noun categories at the level of *ENTITY, STATE, ACT,...*. At level 2 the dictionary maps nouns to all the classes that have a level-1 parent; thus each class can be either a leaf or an intermediate (level 2) class. In general, at level i nouns are only mapped to classes that have a level $(i-1)$, or smaller, parent. There are 34 level-2 classes, 69 level-3 classes and 95 level-4 ones. Finally, at level 5, nouns are mapped to all 106 leaves. We compared the boosting models and the NN and NB classifiers over a fixed size for k of 4.

For each level we extracted all unambiguous instances from the BLLIP'99 data. The data ranged from 200 thousand instances at level 5, to almost 400 thousand at level 1. As the number of classes grows there are less unambiguous words. We randomly selected a fixed number of noun types for each level: 200 types at levels 4 and 5, 300 at level 3, 350 at level 2 and 400 at level 1. Test was limited to common nouns with frequency between 10 and 300 on the total data. No instance of the noun types present in the test set ever appeared in the training data. The test data was between 5 and 10% of the training data; 10 thousand instances at level 5, 16 thousand at level 1, with intermediate figures for the other levels. We used exactly the same partition of the data for all experiments, across all models.

Figure 2 shows the error rate of several simple models at level 1 for increasing values of k. The error keeps dropping until k reaches a value around 4 and then starts rising. Testing for all values of $k \leq 30$ confirmed this pattern. This result suggests that the most useful contextual information is that close to the noun, which should be syntactic-semantic in nature, e.g., predicate-argument preferences. As the window widens, the bag of features becomes more noisy. This fact is not too surprising. If we made the window as wide as the whole document, every noun token in the document would have the same set of features. As expected, as the number of classes increases, the task becomes harder and the error of the classifiers increases. Nonetheless the same general pattern of performance with respect to k holds. As the figure shows NN_{TFIDF} greatly improves over the simpler NN classifier that only uses counts. NB outperforms both.

4 Boosting for word classification

4.1 AdaBoost.MH with abstaining

Boosting is an iterative method for combining the output of many weak classifiers or learners[6] to produce an accurate *ensemble* of classifiers. The method starts with a training set $\mathcal{S}$ and trains the first classifier. At each successive iteration t a new classifier is trained on a new training set $\mathcal{S}_t$, which is obtained by re-weighting the training data used at $t-1$ so that the examples that were misclassified at $t-1$ are given more weight while less weight is given to the correctly classified examples. At each

[6]The learner is called weak because it is required to classify examples better than at random only by an arbitrarily small quantity.

iteration a weak learner $h_t(\cdot)$ is trained and added to the ensemble with weight α_t. The final ensemble has the form

$$F(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x}) \tag{3}$$

In the most popular version of a boosting algorithm, AdaBoost (Schapire and Singer, 1998), at each iteration a classifier is trained to minimize the exponential loss on the weighted training set. The exponential loss is an upper bound on the zero-one loss. AdaBoost minimizes the exponential loss on the training set so that incorrect classification and disagreement between members of the ensemble are penalized.

Boosting has been successfully applied to several problems. Among these is text categorization (Schapire and Singer, 2000), which bears similarities with word classification. For our experiments we used *AdaBoost.MH with real-valued predictions and abstaining*, a version of boosting for multiclass classification described in Schapire and Singer (2000). This version of AdaBoost minimizes a loss function that is an upper bound on the *Hamming distance* between the weak learners' predictions and the real labels, i.e., the number of label mismatches (Schapire and Singer, 1998). This upper bound is the product $\prod_t Z_t$. The function $y_i[l]$ is 1 if l is the correct label for the training example x_i and is -1 otherwise; $d = |\mathcal{Y}|$ is the total number of classes; and $m = |\mathcal{S}|$ is the number of training examples. We explain what the term for the weak learner $h_t^w(x_i, l)$ means in the next section. Then

$$Z_t = \sum_i^m \sum_l^d D_t(i,l) \exp(y_i[l] h_t^w(x_i, l)) \tag{4}$$

AdaBoost.MH looks schematically as follows:

```
ADABOOST.MH($S$)
1  $D_1(x_i, l) \leftarrow \frac{1}{md}$;  ▷ uniform initialization $D_1$
2  for $t \leftarrow 1$ to $t \leftarrow T$
3  do get weak hypothesis $h_t^w$ wrt $D_t$;
4       $D_{t+1}(x_i, l) = \frac{D_t(x_i,l)\exp(-y_i[l]h_t^w(x_i,l))}{Z_t}$;
```

$D(x_i, l)$ is the weight assigned to the instance-label pair (x_i, l). In the first round D each pair is assigned the same weight. At the end of each round the re-weighted D_t is normalized so that it forms a distribution; i.e., Z_t is a normalizing factor. The algorithm outputs the final hypotheses for an instance x_i with respect to class label l

$$f(x_i, l) = \sum_t^T h_t^w(x_i, l) \tag{5}$$

since we are interested in classifying noun types the final score for each unknown noun is

$$F(n, l) = \sum_{i:i \in n} f(x_i, l) \tag{6}$$

where with $i : i \in n$ instance x_i is a token of noun type n.

4.2 Weak learners

In this version of AdaBoost weak learners are extremely simple. Each feature, e.g., one particular collocation, is a weak classifier. At each round one feature w is selected. Each feature makes a real-valued prediction $c_t(w, l)$ with respect to each class l. If $c_t(w, l)$ is positive then feature w makes a positive prediction about class l; if negative, it makes a negative prediction about class l. The magnitude of the prediction $|c_t(w, l)|$ is interpreted as a measure of the confidence in the prediction. Then for each training instance a simple check for the presence or absence of this feature is performed. For example, a possible collocation feature is *eat*:VB, and the corresponding prediction is "if *eat*:VB appears in the context of a noun, predict that the noun belongs to the class *FOOD* and doesn't belong to classes *PLANT, BUSINESS,...*". A weak learner is defined as follows:

$$h_t^w(x_i, l) = \begin{cases} c_t(w, l) & \text{if } w \in x_i \\ 0 & \text{if } w \notin x_i \end{cases} \tag{7}$$

The prediction $c_t(w, l)$ is computed as follows:

$$c_t(w, l) = \frac{1}{2} \ln \left(\frac{W_+^l + \epsilon}{W_-^l + \epsilon} \right) \tag{8}$$

$W_+^l (W_-^l)$ is the sum of the weights of noun-label pairs, from the distribution D_t, where the feature appears and the label is correct (wrong); $\epsilon = \frac{1}{md}$ is a smoothing factor. In Schapire and Singer (1998) it

```
W=August; PL=0; MU=1; CO=’:POS; CO=passenger:NN; CO=traffic:NN; ...
W=punishment; PL=1; MU=0; MS=ment; MS=ishment; CO=in:IN; CO=to:TO; ...
W=vice_president; PL=0; MU=0; MSHH=president; CO=say:VB; CO=chief:JJ; ...
W=newsletter; PL=0; MU=0; MS=er; MSSH=letter; CO=yield:NN; CO=seven-day:JJ; ...
```

Figure 3: Sample input to the classifiers, only $Boost_M$ has access to morphological information. CO stands for the attribute “collocation”.

is shown that Z_t is minimized for a particular feature w by choosing its predictions as described in equation (8). The weight α_t usually associated with the weak classifier (see equation (2)) here is simply set to 1.

If the value in (8) is plugged into (4), Z_t becomes

$$Z_t = W_0 + 2 \sum_{l \in \mathcal{Y}} \sqrt{W_+^l W_-^l} \qquad (9)$$

Therefore to minimize Z_t at each round we choose the feature w for which this value is the smallest. Updating these scores is what takes most of the computation, Collins (2000) describes an efficient version of this algorithm.

4.3 Morphological features

We investigated two boosting models: $Boost_S$, which uses only collocations as features, and $Boost_M$, which uses also a very simple set of morphological features. In $Boost_S$ we used the collocations within a window of $\pm k = 4$, which seemed to be a good value for both the nearest neighbor and the naive Bayes model. However, we didn’t focus on any method for choosing k, since we believe that the collocational features we used only approximate more complex ones that need specific investigation. Our main goal was to compare models with and without morphological information. To specify the morphological properties of the nouns being classified, we used the following set of features:

- plural (PL): if the token occurs in the plural form, PL=1; otherwise PL=0
- upper case (MU): if the token’s first character is upper-cased MU=1; otherwise MU=0
- suffixes (MS): each token can have 0, 1, or more of a given set of suffixes, e.g., *-er, -ishment, -ity, -ism, -esse, ...*
- prefixes (MP): each token can have 0, 1 or more prefixes, e.g., *pro-, re-, di-, tri-, ...*
- Words that have complex morphology share the morphological head word if this is a noun in Wordnet. There are two cases, depending on whether the word is hyphenated (MSHH) or the head word is a suffix (MSSH)
  - hyphenated (MSHH): *drinking_age* and *age* share the same head-word *age*
  - non-hyphenated (MSSH): *chairman* and *man* share the same suffix head word, *man*. We limited the use of this feature to the case in which the remaining prefix (*chair*) also is a noun in Wordnet.

We manually encoded two lists of 61 suffixes and 26 prefixes[7]. Figure 3 shows a few examples of the input to the models. Each line is a training instance; the attribute W refers to the lexical form of the noun and was ignored by the classifier.

4.4 Stopping criterion

One issue when using iterative procedures is deciding when to stop. We used the simplest procedure of fixing in advance the number of iterations. We noticed that the test error drops until it reaches a point at which it seems not to improve anymore. Then the error oscillates around the same value even for thousands of iterations, without apparent overtraining. A similar behavior is observable in some of the results on text categorization presented in (Schapire and Singer, 2000). We cannot say that overtraining is not a potential danger in multiclass boosting models. However, for our experiments, in which the main goal is to investigate the impact of a particular class of features, we could limit the number of

[7]The feature lists are available together with the MiniWordnet files.

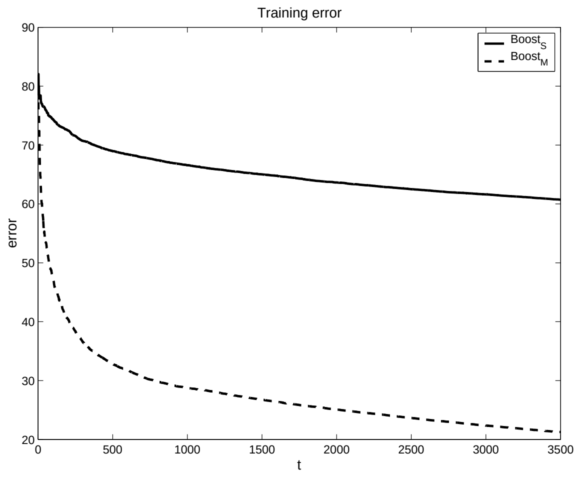

Figure 4: Training error at level 4.

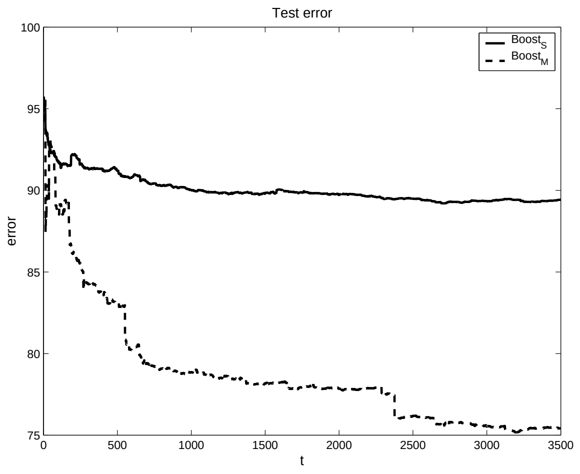

Figure 5: Test error at level 4.

iterations to a fixed value for all models. We chose this maximum number of iterations to be 3500; this allowed us to perform the experiments in a reasonable time. Figure 4 and Figure 5 plot training and test error for $Boost_S$ and $Boost_M$ at level 4 (per instance). As the figures show, the error rate, on both training and testing, is still dropping after the fixed number of iterations. For the simplest model, $Boost_S$ at level 1, the situation is slightly different: the model converges on its final test error rate after roughly 200 iterations and then remains stable. In general, as the number of classes grows, the model takes more iterations to converge and then the test error remains stable while the training error keeps slowly decreasing.

5 Results and discussion

The following table summarizes the different models we tested:

MODEL	FEATURES
NN_{TFIDF}	TFIDF weights for collocations
NB	collocation counts
Boost_s	collocations (binary)
Boost_m	collocations (binary)+morphology

Figure 6 plots the results across the five different subsets of the reduced lexicon. The error rate is the error on types. We also plot the results of a baseline (BASE), which always chooses the most frequent class and the error rate for random choice (RAND). The baseline strategy is quite successful on the first sets of classes, because the hierarchy under the root $ENTITY$ is by far the most populated. At level 1 it performs worse only than $Boost_M$. As the size of the model increases, the distribution of classes becomes more uniform and the task becomes harder for the baseline. As the figure shows the impact of morphological features is quite impressive. The average decrease in type error of $Boost_M$ over $Boost_S$ is more than 17%, notice also the difference in test and training error, per instance, in Figures 4 and 5.

In general, we observed that it is harder for all classifiers to classify nouns that don't belong to the ENTITY class, i.e., maybe not surprisingly, it is harder to classify nouns that refer to abstract concepts such as *groups*, *acts*, or *psychological features*. Usually most of the correct guesses regard members of the ENTITY class or its descendants, which are also typically the classes for which there is more training data. $Boost_M$ really improves on $Boost_S$ in this respect. $Boost_M$ guesses correctly several nouns to which morphological features apply like *spending*, *enforcement*, *participation*, *competitiveness*, *credibility* or *consulting_firm*. It makes also many mistakes, for example on *conversation*, *controversy* and *insurance_company*. One problem that we noticed is that there are several cases of nouns that have intuitively meaningful suffixes or prefixes that are not present in our hand-coded lists. A possible solution to his problem might be the use of more general morphological rules like those used in part-of-speech tagging models (e.g.,

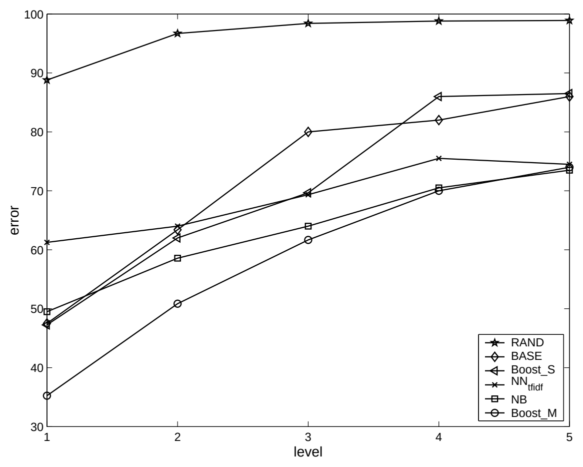

Figure 6: Comparison of all models for $l = 1..5$.

Ratnaparkhi (1996)), where all suffixes up to a certain length are included. We observed also cases of recurrent confusion between classes. For example between *ACT* and *ABSTRACTION* (or their subordinates), e.g., for the noun *modernization*, possibly because the suffix is common in both cases.

Another measure of the importance of morphological features is the ratio of their use with respect to that of collocations. In the first 100 rounds of $Boost_M$, at level 5, 77% of the features selected are morphological, 69% in the first 200 rounds. As Figures 4 and 5 show these early rounds are usually the ones in which most of the error is reduced. The first ten features selected at level 5 by $Boost_M$ were the following: PL=0, MU=0, PL=1, MU=0, PL=1, MU=1, MS=ing, PL=0, MS=tion, and finally CO=NUMBER_X:CD. One final characteristic of morphology that is worth mentioning is that it is independent from frequency. Morphological features are properties of the type and not just of the token. A model that includes morphological information should therefore suffer less from sparse data problems.

From a more general perspective, Figure 6 shows that even if the simpler boosting model's performance degrades more than the competitors after level 3, $Boost_M$ performs better than all the other classifiers until level 5 when the TFIDF nearest neighbor and the naive Bayes classifiers catch up. It should be noted though that, as Figures 4 and 5 showed, boosting was still improving at the end of the fixed number of iterations at level 4 (but also 5). It might quite well improve significantly after more iterations. However, determining absolute performance was beyond the scope of this paper. It is also fair to say that both NN and NB are very competitive methods, and much simpler to implement efficiently than boosting. The main advantage with boosting algorithms is the flexibility in managing features of very different nature. Feature combination can be performed naturally with probabilistic models too but it is more complicated. However, this is something worth investigating.

6 Related work

Automatic lexical acquisition is a classic problem in AI. It was originally approached in the context of story understanding with the aim of enabling systems to deal with unknown words while processing text or spoken input. These systems would typically rely heavily on script-based knowledge resources. FOUL-UP (Granger, 1977) is one of these early models that tries to deterministically maximize the expectations built into its knowledge base. Jacobs and Zernik (1988) introduced the idea of using morphological information, together with other sources, to guess the meaning of unknown words. Hastings and Lytinen (1994) investigated attacking the lexical acquisition problem with a system that relies mainly on taxonomic information. In the last decade or so research on lexical semantics has focused more on sub-problems like word sense disambiguation (Yarowsky, 1995; Stevenson and Wilks, 2001), named entity recognition (Collins and Singer, 1999), and vocabulary construction for information extraction (Riloff, 1996). All of these can be seen as sub-tasks, because the space of possible classes for each word is restricted. In WSD the possible classes for a word are its possible senses; in named entity recognition or IE the number of classes is limited to the fixed (usually small) number the task focuses on. Other kinds of models that have been studied in the context of lexical acquisition are those based on lexico-syntactic patterns of the kind ”X, Y and other Zs”, as in the phrase ”bluejays, robins and other birds”. These types of models have been used for hyponym discovery (Hearst, 1992; Roark and Charniak, 1998), meronym discovery (Berland and Charniak, 1999), and hierarchy building (Caraballo, 1999). These methods are very interesting but of limited applicability, because nouns that do not appear in known lexico-syntactic patterns cannot be learned.

7 Conclusion

All the approaches cited above focus on some aspect of the problem of lexical acquisition. What we learn from them is that information about the meaning of words comes in very different forms. One thing that needs to be investigated is the design of better sets of features that encode the information that has been found useful in these studies. For example, it is known from work in word sense disambiguation that conditioning on distance and syntactic relations can be very helpful. For a model for lexical acquisition to be successful it must be able to combine as many sources of information as possible. We found that boosting is a viable method in this respect. In particular, in this paper we showed that morphology is one very useful source of information, independent of frequency, that can be easily encoded in simple features.

A more general finding was that inserting new words into a dictionary is a hard task. For these classifiers to become useful in practice, much better accuracy is needed. This raises the question of the scalability of machine learning methods to multiclass classification for very large lexicons. Our impression on this is that directly attempting classification on tens of thousands of classes is not a viable approach. However, there is a great deal of information in the structure of a lexicon like Wordnet. Our guess is that the ability to make use of structural information will be key in successful approaches to this problem.

References

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

S. Caraballo. 1999. Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th ICML*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

R. Granger. 1977. Foul-up: A program that figures out meanings of words from context. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*.

P.M. Hastings and S.L. Lytinen. 1994. The ups and downs of lexical acquisition. In *AAAI-94*.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*.

P. Jacobs and U. Zernik. 1988. Acquiring lexical knowledge from text: A case study. In *AAAI-88*.

G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1990. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4).

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the First Empirical Methods in Natural Language Processing Conference*.

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

B. Roark and E. Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*.

R. E. Schapire and Y. Singer. 1998. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*.

R. E. Schapire and Y. Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39.

M. Stevenson and Y. Wilks. 2001. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics*, 27.

Y. Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.