Hierarchical Semantic Classification: Word Sense Disambiguation with World Knowledge*

Massimiliano Ciaramita
Brown University
massi@brown.edu

Thomas Hofmann
Brown University
th@cs.brown.edu

Mark Johnson
Brown University
mark_johnson@brown.edu

Abstract

We present a learning architecture for lexical semantic classification problems that supplements task-specific training data with background data encoding general "world knowledge". The model compiles knowledge contained in a dictionary-ontology into additional training data, and integrates task-specific and background data through a novel hierarchical learning architecture. Experiments on a word sense disambiguation task provide empirical evidence that this "hierarchical classifier" outperforms a state-of-the-art standard "flat" one.

1 Introduction

There is an increasing interest in natural language processing (NLP) and information retrieval (IR) for research on lexical semantics, in particular with respect to word sense disambiguation [Yoong and Hwee, 2002], information extraction [Riloff and Jones, 1999], named entity recognition [Collins, 2002], and automatic thesaurus extension [Hearst, 1992]. In general terms, the goal in these tasks is that of automatically associating words in text with semantic labels. In information extraction and named-entity recognition noun phrases or proper nouns are assigned to semantic categories such as "organization", "person", or "location". In word sense disambiguation and thesaurus extension the goal is to assign words to finer-grained categories defined by existing dictionaries and ontologies.

Lexical semantic information can be useful in many NLP and IR applications such as text categorization, parsing, and language modeling for speech recognition. Furthermore it can be crucial for tasks that require complex inferences involving world knowledge, such as question answering.

One of the main difficulties in learning semantic annotations stems from the fact that training instances are often narrowly focused on very specific class labels and relatively few in number. It thus seems intuitive to supplement task-specific training data, for example, sense-annotated training instances for a specific word, with background data encoding general "world knowledge". The latter are typically available in sufficient quantities and need not to be generated separately for each classification task. To carry out this idea two crucial issues need to be addressed: How exactly can world knowledge be compiled into additional training data, and how can task-specific and background data be systematically integrated?

To address the first challenge, we propose to generate additional training data about broader semantic categories by extracting training sentences from a hierarchically structured ontology, WordNet[1] [Fellbaum, 1998]. We assumed that each example sentence associated with a lexical entry provides evidence for the kind of contexts in which that specific concept and all its ancestors in the hierarchy can appear. As far as the second challenge is concerned, we introduce a novel hierarchical learning architecture for semantic classification. More specifically, we present a simple and efficient on-line training algorithm generalizing the multiclass perceptron of [Crammer and Singer, 2002].

Finally, we carry out an experimental evaluation on a word sense disambiguation task, providing empirical evidence that the hierarchical classifier outperforms a state-of-the-art standard "flat" classifier for this task.

The paper is structured as follows. Section 2 introduces the main idea in more detail. In Section 3 we introduce WordNet and the simplified ontology derived from it that we used as the source of world knowledge. Section 4 deals with the basic multiclass perceptron and the proposed hierarchical multicomponent classifier. Finally, Sections 5 and 6 describe the data set used and the empirical results, respectively.

2 Word Sense Disambiguation and World Knowledge

Word sense disambiguation is the task of assigning to each occurrence of an ambiguous word in a text one of its possible senses. A dictionary is used to decide if a lexical entry is ambiguous or not, and to specify its set of possible senses. The most widely used lexical resource for this task is WordNet, which we describe in detail in the next section.

*We would like to thank our colleagues in the Information Retrieval and Machine Learning Group (IRML) and Brown Laboratory for Linguistic Information Processing (BLLIP), as well as Jesse Hochstadt for his editing advice. This material is based upon work supported by the National Science Foundation under Grant No. 0085940.

[1] In this paper we always refer to WordNet version 1.71.

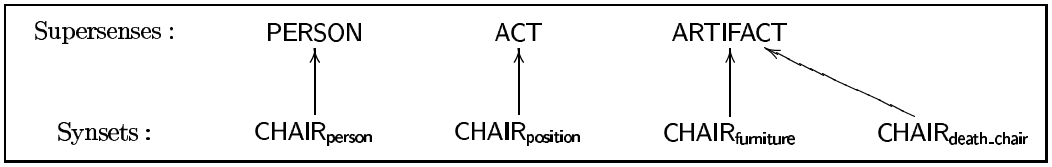

Figure 1. The simplified two-layer hierarchy for the noun *chair*.

As an illustration consider the noun "chair", which according to WordNet is ambiguous. Two possible senses are explained in the following WordNet entries:

- $chair_1$ – a seat for one person, with a support for the back;
- $chair_2$ – (president, chairman, chairwoman, chair, chairperson), the officer who presides at the meetings of an organization;

Word sense disambiguation is often framed as a multi-class pattern classification task. Useful features include co-occurring words, word bigrams or trigrams, and properties of the syntactic context that contains the target word. Most commonly systems are trained on labeled data for a specific word, for each and tested on unseen items of the same word. The set of possible labels is the set of senses of the ambiguous word. One limitation of such a strategy is that the system bases its decision exclusively on what it has been able to learn about a few very specific concepts; e.g., $chair_1$ and $chair_2$. Furthermore, since manually sense-tagging words for the required training data is slow and expensive, the data is quite sparse.

A great deal of information about objects like "chairs" is indirect and can be derived from more general world knowledge through generalization and inference processes. Suppose that the task is to disambiguate between the two simple senses of *chair* in the following context:

1) "Here the quality of the finest *chair* components is merged with art."

In this sentence *components* is a useful hint that we are dealing with the sense $chair_1$. Chairs are artifacts, and artifacts can have components. Conversely, even though in principle people could "have components" as well, this sounds a little odd. Intuitively, if a word sense disambiguation system had access to this type of information - that "chairs" are subordinates of broader concepts like "artifacts" and "people" - and some knowledge about these broader semantic categories, it might achieve a higher accuracy in disambiguating words. Notice that the system might never have previously observed any instance of the noun "chair", in either sense, as "having components".

The goal hence is to complement specific but limited knowledge about narrow classes with richer, if less specific, knowledge about more general classes. We can easily recover the fact that chairs are kinds of furniture or people from dictionaries and hierarchically organized ontologies like WordNet. Learning information about such general concepts, however, is complicated. One source of complication is the very problem we are trying to solve, lexical ambiguity. If we do not know whether something is a person or an artifact we cannot learn reliable information about those more general concepts. One way of addressing these problems is offered by WordNet itself.

3 The Ontology

3.1 WordNet

WordNet is a broad-coverage, machine-readable dictionary widely used in NLP. The English version contains around 150,000 entries, mostly nouns, but also verbs, adjectives, and adverbs. WordNet is organized as a network of lexicalized concepts, called *synsets*, that comprise sets of synonyms. For example, the nouns {president, chairman, chairwoman, chair, chairperson} form a synset. A word that belongs to several synsets is *ambiguous*. Synsets are linked by semantic relations, the most important of which for nouns and verbs is the is-a relation, or *hyponymy*; e.g., "car" is a hyponym of "vehicle". The verb and noun databases form is-a hierarchies with a few general concepts at the top and several thousand specific concepts at the leaf level.

The hierarchical structure of the database has aroused some interest in NLP, because it can support interesting computational language learning models, for example, in learning predicate selectional preferences [Light and Greif, 2002]. We aim to use the hierarchy to improve lexical classification methods. The model we present here can in principle make use of the full hierarchy. However, for the sake of simplicity we have focused on a less complex hierarchy, which has been derived from WordNet as described below.

3.2 A simple two-level hierarchy

WordNet was built, and is regularly updated, by lexicographers. Lexicographers group words together in synsets and individuate the relevant semantic relations between synsets. This process includes the classification of lexical entries into one of 26 broad semantic classes. In this paper we refer to these broad classes with the term *supersenses*. A few examples of supersense labels are person, animal, artifact, food, location, time, plant, process, attribute, substance, and relation. This set of labels is fairly general and therefore small. At the same time the labels are not too abstract. In other words, these classes seem natural and easily recognizable, and that is probably why lexicographers use them. In fact the level of generality is very close to that used in named-entity recognition ("location", "person", "organization", etc.).

Each synset in WordNet is associated with one supersense label. As a result the database implicitly defines, in addition to the full hierarchy, a simpler two-layer hierarchy. Figure

1 above illustrates the synsets and supersenses *chair* belongs to.

3.3 The hierarchy as a source of world knowledge

For a few thousand concepts WordNet lists, among other types of semantic information, one or more example sentences. For the sense of chair above the example sentences are the following:

- $chair_1$ – "he put his coat over the back of the chair and sat down"
- $chair_2$ – "address your remarks to the chairperson"

Overall there are 9,258 of these sentences. Since each one is associated with one synset, that is in fact a sense-tagged instance of the word. In other words, WordNet provides a few thousand potential sense-tagged training instances.

Unfortunately, this additional data in itself would not be of much help: for most of the synsets there are no sentences[2], and typically the sentences are very short and do not provide much context. However, the situation appears in a different light if we take into account the hierarchy. Considering an example sentence for a synset also as an example sentence for its ancestors (synsets at higher levels in the hierarchy), the number of sentences grows larger at the superordinate levels. If we consider the supersense level, the set of example sentences constitutes in fact a small corpus of supersense-annotated data. Our hypothesis is that the several hundred sentences associated with each supersense can provide a useful source of general world knowledge. In the next section we describe a general multicomponent learning architecture that can be used to exploit this supplementary training data.

4 Multicomponent Learning Architecture

The idea of using the hierarchical structure of a domain to overcome sparseness problems has been explored in text categorization. These methods show improved accuracy and efficiency [Toutanova et al., 2001; Dumais and Chen, 2000]. In NLP the hierarchical structure of WordNet has been used to overcome sparseness data problems for estimating class distributions [Clark and Weir, 2002], and to exploit morphological information to improve lexical acquisition [Ciaramita, 2002].

4.1 Multiclass perceptron

The architecture we propose is a generalization of "ultraconservative" on-line learning [Crammer and Singer, 2002], which is itself an extension of perceptron learning to the multiclass case. We describe this "flat" version of the classifier first. For each noun w we are given a training set $S = (x_i, y_i)_{i=1}^n$, where each instance $x_i \in \mathbb{R}^d$ and $y_j \in Y(w)$. $Y(w)$ is the set of synsets that WordNet assigns to w. Thus S summarizes n instances of noun w, where each instance i is represented as a vector of features x_i extracted from the context in which w occurred; d is the total number of features and y_i is the true label of x_i.[3]

[2]There are in total around 75,000 synsets in the noun database.

[3]Since some instances are labeled with multiple senses, in cases where the taggers were uncertain, y_i may actually be a set of labels.

Algorithm 1 Multiclass Perceptron

input training data $(x_i, y_i)_{i=1}^n$, $\mathbf{V} = 0$
repeat
  for $i = 1, ..., n$ **do**
    if $H_w(x_i; \mathbf{V}) \neq y_i$ **then**
      $v_{y_i} \leftarrow v_{y_i} + x_i$
      $E_i = \{y \in Y(w) : \langle v_y, x_i \rangle > \langle v_{y_i}, x_i \rangle\}$
      for $y \in E_i$ **do**
        $v_y \leftarrow v_y - \frac{1}{|E_i|} x_i$
      end for
    end if
  end for
until no more mistakes

In general, a multiclass classifier for word w is a function $H_w : \mathbb{R}^n \rightarrow Y(w)$ that maps feature vectors x to one of the possible senses of w. In the multiclass perceptron, one introduces a weight vector $v_y \in \mathbb{R}^d$ for every $y \in Y(w)$ and defines H_w implicitly by the so-called winner-take-all rule:

$$H_w(x; \mathbf{V}) = \arg \max_{y \in Y(w)} \langle v_y, x \rangle . \quad (1)$$

Here $\mathbf{V} \in \mathbb{R}^{k \times d}$ refers to the matrix of weights, every column corresponding to one of the weight vectors v_y.

The learning algorithm works as follows: Training patterns are presented one at a time in the standard on-line learning setting. Whenever $H_w(x_i; \mathbf{V}) \neq y_i$ an update step is performed; otherwise the weight vectors remain unchanged. To perform the update, one first computes the error set E_i containing those class labels that have received a higher score than the correct class:

$$E_i = \{y \in Y(w) : \langle v_y, x_i \rangle > \langle v_{y_i}, x_i \rangle\} \quad (2)$$

An ultraconservative update scheme in its most general form is then defined as follows: Update $v_y \leftarrow v_y + \tau_y x_i$ with learning rates fulfilling the constraints $\tau_{y_i} = 1$, $\sum_{y \neq y_i} \tau_y = -1$, and $\tau_y = 0$ for $y \notin E_i \cup \{y_i\}$. Hence changes are limited to v_y for $y \in E_i \cup \{y_i\}$. The sum constraint ensures that the update is balanced, which is crucial to guaranteeing the convergence of the learning procedure (cf. [Crammer and Singer, 2002]). We have focused on the simplest case of uniform update weights, $\tau_y = -\frac{1}{|E_i|}$ for $y \in E_i$. The algorithm is summarized in Algorithm 1.

Notice that the presented multiclass perceptron algorithm learns all weight vectors in a coupled manner, in contrast to methods that perform multiclass classification by combining binary classifiers, for example, training a classifier for each class in a one-against-the-rest manner.

4.2 Hierarchical multiclass perceptron

The hierarchical multiclass perceptron is inspired by the framework for learning over structured output spaces introduced in [Hofmann et al., 2002]. The key idea is to introduce a weight vector not only for every (leaf-level) class, but also for every inner node in a given class taxonomy. In the current application to word sense disambiguation, the inner nodes correspond to the 26 supersenses s and we will hence

introduce additional weight vectors v_s for $s \in S(w)$, where $S(w)$ refers to the subset of supersenses induced by $Y(w)$. We will use the notation $s(y)$ to refer to the supersense corresponding to a synset y. Then discriminant functions $f(x, y)$ can be defined in an additive manner by

$$f(x, y; \mathbf{V}) = \langle v_y, x \rangle + \langle v_{s(y)}, x \rangle . \quad (3)$$

If one thinks of f in terms of a compatibility function between an observation vector x and a synset y, then the compatibility score is simply the sum of two independent contributions, one stemming from the supersense level and the other one coming from the more detailed synset level. The multiclass classifier is then again defined using the winner-take-all rule,

$$H_w(x; \mathbf{V}) = \arg\max_y f(x, y; \mathbf{V}), \quad (4)$$

Algorithm 2 Hierarchical Multiclass Perceptron

1: **input** training data $(x_j, s_j)_{j=1}^m$ and $(x_i, y_i)_{i=1}^n$, $\mathbf{V} = 0$
2: **repeat**
3: **for** $j = 1, ..., m$ **do**
4: $E_j^S = \{s \in S(w) : \langle v_s, x_j \rangle > \langle v_{s_j}, x_j \rangle\}$
5: **if** $E_j^S \neq \emptyset$ **then**
6: $v_{s_j} \leftarrow v_{s_j} + x_j$
7: **for** $s \in E_j^S$ **do**
8: $v_s \leftarrow v_s - \frac{1}{|E_j^S|} x_j$
9: **end for**
10: **end if**
11: **end for**
12: **for** $i = 1, ..., n$ **do**
13: **if** $H_w(x_i; \mathbf{V}) \neq y_i$ **then**
14: $v_{y_i} \leftarrow v_{y_i} + x_i$
15: $E_i = \{y \in Y(w) : f(x, y; \mathbf{V}) > f(x, y_i; \mathbf{V})\}$
16: **for** $y \in E_i$ **do**
17: $v_y \leftarrow v_y - \frac{1}{|E_i|} x_i$
18: **end for**
19: $E_i^S = \{s = s(y) : f(x, y; \mathbf{V}) > f(x, y_i; \mathbf{V})\}$
20: **if** $E_i^S \neq \emptyset$ **then**
21: $v_{s_i} \leftarrow v_{s_i} + x_i$
22: **for** $s \in E_i^S$ **do**
23: $v_s \leftarrow v_s - \frac{1}{|E_i^S|} x_i$
24: **end for**
25: **end if**
26: **end if**
27: **end for**
28: **until** no more mistakes

The complete algorithm is summarized in Algorithm 2. The first part of the algorithm concerns the different nature of the two types of training data. As we explained in Section 3, the supplementary data derived from WordNet only provides annotations at the supersense level. We cannot use this information to perform updates for weight vectors v_y, but only to adjust the weights v_s. Hence for supersense-annotated training instances (x_j, s_j) we compute the error set on the supersense level as $E_j^S = \{s : \langle v_s, x_j \rangle > \langle v_{s_j}, x_j \rangle\}$ and perform the standard multiclass update step for all v_s with $s \in E_j^S \cup \{s_j\}$.

The second part concerns training on the task-specific data (x_i, s_i). If the classifier makes a mistake on pattern x_i, error sets are computed for its individual components both at the synset and supersense levels (lines 15 and 19 above), which are updated according to the standard multiclass update rule.

As an example, suppose that given a pattern x_i of *chair*, the synset error set is $E_i = \{\mathsf{CHAIR}_{\mathsf{furniture}}, \mathsf{CHAIR}_{\mathsf{death_chair}}\}$, while the correct label is $y_i = \mathsf{CHAIR}_{\mathsf{person}}$, and thus $s_i = \mathsf{PERSON}$. The update vector $(\frac{1}{2})x_i$ is subtracted from the vectors relative to the labels in E_i, while x_i is added to v_{y_i}. If at the supersense level the error set $E_i^S = \{\mathsf{ARTIFACT}\}$, x_i is subtracted from the vector for $\mathsf{ARTIFACT}$ and added to v_{s_i}. Therefore, through the supersense weight vectors, the background data affects classification at the synset level.

5 Data Set and Features

5.1 The Senseval data

We tested our system on a standard word sense disambiguation data set. The training and test data are those used in the last Senseval workshop (Senseval-2/ACL-01, 2001), which focused exclusively on word sense disambiguation. The training set consists of 8,611 paragraphs that contain an ambiguous word whose sense has been manually annotated. The inventory of senses is taken from WordNet. Similarly, the test set consists of 4,328 unlabeled pairs. We only ran experiments on the noun data, which consists of 3,512 training instances and 1,754 test instances. Each instance consists of a short passage taken from one of various sources: e.g., the Wall Street Journal, British National Corpus, and web pages. The task-specific training data, T_Y, is typically smaller than the general one, T_S. The average ratio $|T_S|/|T_Y|$ is equal to 20.3.

5.2 Features

We used the same feature set described in [Yoong and Hwee, 2002], which is compact but includes most of the features that have been found useful in this task: surrounding words, bigrams and trigrams, and syntactic information. Yoong and Hwee report results for several classifiers broken down by part of speech, which makes it possible to compare our system's performance with that of several others.

There are four types of features. The following sentence serves to illustrate them: "the dinner table and *chairs* are elegant yet comfortable". The feature set is described in greater details in [Yoong and Hwee, 2002]:

- part of speech of the neighboring words: $\mathtt{P}_{-1} = \mathtt{CC}$, $\mathtt{P}_0 = \mathtt{NNS}$, $\mathtt{P}_{+1} = \mathtt{AUX}$, ...
- single words in the surrounding context: $\mathtt{C} = \mathtt{elegant}$, $\mathtt{C} = \mathtt{dinner}$, $\mathtt{C} = \mathtt{table}$, $\mathtt{C} = \mathtt{the}$, ...
- bigrams and trigrams: $\mathtt{C}_{-1,+1} = \mathtt{and_are}$, $\mathtt{C}_{-2,-2} = \mathtt{table}$, $\mathtt{C}_{+1,+2} = \mathtt{are_elegant}$, ...
- head of the syntactic phrase that governs the target: $\mathtt{G} = \mathtt{are}$, $\mathtt{G_POS} = \mathtt{AUX}$, $\mathtt{G_VOICE} = \mathtt{active}$, $\mathtt{G_RELPOS} = \mathtt{left}$.

Syntactic features and part of speech tags were extracted from the syntactic parse trees of the Senseval-2 training and

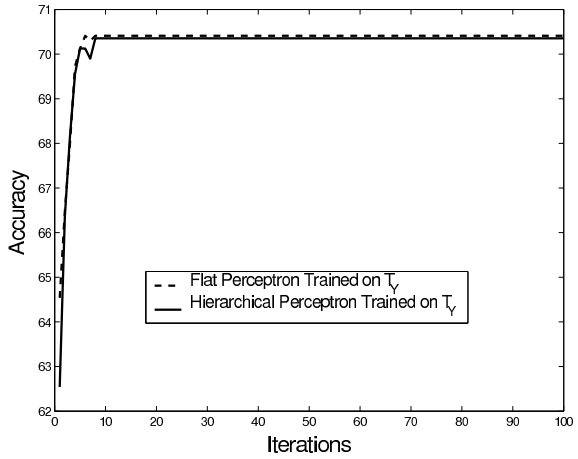

Figure 2. Test accuracy of the flat multiclass perceptron (dashed line) and the hierarchical multiclass perceptron (continuous line) on the word sense evaluation data set.

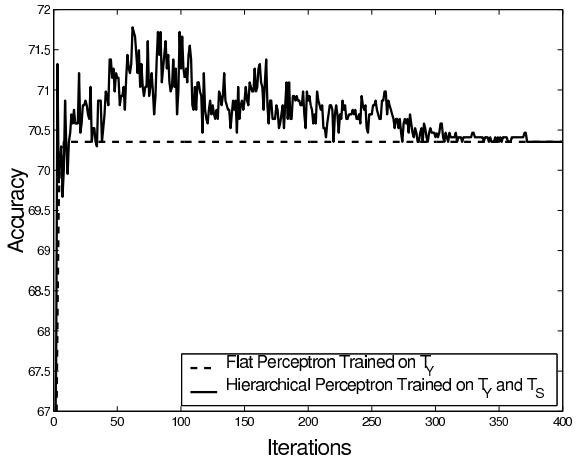

Figure 3. Test accuracy of the hierarchical (continuous line) vs. flat (dashed line) multiclass perceptron. The hierarchical multiclass perceptron was trained using supplementary supersense training data.

test data produced using Charniak's parser [Charniak, 2000]. In this way we created the training data T_Y from the Senseval data. In exactly the same way we extracted features from the example sentences in WordNet to produce the additional training set for the supersense-level classes, T_S. Overall there are around 250,000 features.

6 Experiments

6.1 Experimental setup

We tested two models described in Section 4: the flat multiclass perceptron, trained and tested at the synset level, and the hierarchical one, trained on both the standard synset data and the training data for the supersenses extracted from WordNet. We also trained and tested a simple "flat" naive Bayes classifier. A different classifier was trained and tested for each word. We treated compounds such as *easy chair* and *chair* as different words.

All the results we report are given as accuracy:

$$\text{score} = \frac{\text{number of correct guesses}}{\text{number of test items}} \times 100$$

.

6.2 Results

Figure 2 shows the performance of the flat perceptron (dotted line) during each iteration. The perceptron in fact converges very quickly. This is probably due to the fact that there are relatively few training items: Normally the size of T_Y is between one and two hundred. To check whether an improvement was due to the algorithm alone and not to the combination of the algorithm and the additional, supersense data set, we also trained a hierarchical perceptron exclusively on the synset data. Figure 2 also plots the performance of the hierarchical perceptron trained only on T_Y. The two curves are virtually indistinguishable, meaning that without additional information not much can be gained from using the hierarchical classifier alone. In other words, with "flat" data a "flat" classifier is as good as a "hierarchical" one.

Figure 3 plots the performances of the flat and the hierarchical perceptron when also trained on T_S. The two patterns are very different. The hierarchical model converges only after more than 350 iterations.

Method	Score
Yoong's AdaBoost	69.2
Best S2	69.5
Naive Bayes	68.0
Multiclass perceptron	70.4
Hierarchical multiclass perceptron	71.8

Table 1. Test accuracy on the Senseval-2 test data.

This might be due to several facts. First, the amount of data is much greater due to the addition of T_S, and it takes longer to learn. Second, the supersense data and the synset data are probably very different and noisy; as a consequence the weight vectors are continually readjusted, possibly along very different dimensions. The interesting thing, though, is that even in the midst of very wide oscillations there is a clear improvement, particularly between 50 and 100 iterations.

We also present a comparative table. Table 1 illustrates the results of our systems and other state-of-the-art word sense disambiguation systems. We set the number of iterations to 100 for all words. Given that we set this value "knowing" that it is a good one for both our systems, our results and those of other systems are not really comparable. However, it is reasonable to expect that it is possible to set this stopping criterion well enough using held-out data. Thus this comparison gives us an approximate idea of where our systems stand with respect to state-of-the-art ones in terms of performance. AdaBoost is the classifier that gave the best result on nouns in [Yoong and Hwee, 2002], Best S2 [Mihalcea and Moldovan, 2001] refers to the best-performing system on nouns among the Senseval-2 workshop systems. These results show that our systems' performance is comparable to

noun	F	H	noun	F	H
stress	48.7	53.9	child	63.5	65.1
church	73.8	76.9	bar	73.4	69.1
mouth	65.5	69.0	day	68.4	69.1
sense	59.0	64.1	post	45.6	51.5
art	61.2	57.6	fatigue	81.0	88.1
chair	81.9	83.3	bum	73.3	80.0

Table 2. Example results on a few words. F = flat, H = hierarchical.

that of state-of-the-art ones and that our hierarchical model trained on background and specific data outperforms the flat one.

Results for a few individual words are presented in Table 2. They show that the improvements are not uniform, but vary from word to word. Overall we identified 105 nouns. The great majority of these are compounds that typically occur only once in the test data. Both systems achieve approximately the same score on these data. On the bulk of the test data, however, the systems perform differently. Of the 21 test words on which the classifiers achieve different scores, the hierarchical perceptron is more accurate than the flat one on 15 words, or 71.5% of the time.

This finding suggests a simple improvement for the hierarchical system. The contribution of the individual components of the classifier could be weighted setting the weights, after training, using held-out data. In the simplest setting binary weights could be used; e.g., either the background information is used or not. Thus the background model would be used only when useful, otherwise its contributions would be ignored.

7 Conclusion

We have presented a learning architecture for lexical semantic classification that supplements task-specific training data with background data encoding general "world knowledge" extracted from a widely used broad-coverage, machine-readable dictionary. The model integrates task-specific and general information through a novel hierarchical learning architecture based on the multiclass perceptron. Experiments on a word sense disambiguation task showed that the hierarchical model achieves improved performance over a state-of-the-art standard "flat" system.

This new framework has a number of promising extensions. Additional accuracy gains are expected by using more sophisticated perceptron learning algorithms such as the voted perceptron [Freund and Schapire, 1998] and by using the dual perceptron with non-linear kernels. We have only made use of the simplest possible form of hierarchy (two-stage), in reality the hierarchical structure of WordNet is very complex and much more informative. The model presented here can be extended to include this type of structure as well as other sources of information. In addition, the two-layer model can be applied to all other open-class words in WordNet and a full-hierarchy-based model could be applied to verbs and nouns. There is also more information to extract from WordNet, for example, from the glosses, which can potentially be utilized as additional training data. Lastly, the ideas we presented here might be used with other learning methods. We leave these topics for future research.

References

[Charniak, 2000] E. Charniak. A Maximum-Entropy-Inspired Parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL-00*, 2000.

[Ciaramita, 2002] M. Ciaramita. Boosting Automatic Lexical Acquisition with Morphological Information. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition, ACL-02*, 2002.

[Clark and Weir, 2002] S. Clark and D. Weir. *Class-Based Probability Estimation using a Semantic Hierarchy*. Computational Linguistics, 28, 2002.

[Collins, 2002] M. Collins. Ranking Algorithms for Named-Entity Extraction: Boosting and the Voted Perceptron. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics, ACL-02*, 2002.

[Crammer and Singer, 2002] K. Crammer and Y. Singer. Ultraconservative On line Algorithms for Multiclass Problems. Technical Report [2001-18], School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel, 2002.

[Dumais and Chen, 2000] S. Dumais and H. Chen. Hierarchical Classifi cation of Web Content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, 2000.

[Fellbaum, 1998] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts, 1998.

[Freund and Schapire, 1998] Y. Freund and R.E. Schapire. *Large Margin Classification Using the Perceptron Algorithm.* Machine Learning, 37, 1999.

[Hearst, 1992] M. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, 1992.

[Hofmann et al., 2002] T. Hofmann, I. Tsochantaridis and Y. Altun. Learning over Discrete Output Spaces via Joint Kernel Functions. In *Advances in Neural Information Processing Systems, Workshop on Kernel Methods*, 2002.

[Light and Greif, 2002] M. Light and W. Greiff. *Statistical Models for the Induction and Use of Selectional Preferences*. In Cognitive Science, 87, 2002.

[Mihalcea and Moldovan, 2001] R. Mihalcea and D.I. Moldovan. Pattern Learning and Automatic Feature Selection for Word Sense Disambiguation. In *Proceedings of the Second international Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, 2001.

[Riloff and Jones, 1999] E. Riloff and R. Jones. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

[Toutanova et al., 2001] K. Toutanova, F. Chen, K. Popat and T. Hofmann. Text Classifi cation in a Hierarchical Mixture Model for Small Training Sets. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, 2001.

[Yoong and Hwee, 2002] K.L Yoong and T.N. Hwee. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, 2002.