Efficient Hierarchical Entity Classifier Using Conditional Random Fields

Koen Deschacht
Interdisciplinary Centre for Law & IT
Katholieke Universiteit Leuven
Tiensestraat 41, 3000 Leuven, Belgium
koen.deschacht@law.kuleuven.ac.be

Marie-Francine Moens
Interdisciplinary Centre for Law & IT
Katholieke Universiteit Leuven
Tiensestraat 41, 3000 Leuven, Belgium
marie-france.moens@law.kuleuven.be

Abstract

In this paper we develop an automatic classifier for a very large set of labels, the WordNet synsets. We employ Conditional Random Fields (CRFs) because of their flexibility to include a wide variety of non-independent features. Training CRFs on a big number of labels proved a problem because of the large training cost. By taking into account the hypernym/hyponym relation between synsets in WordNet, we reduced the complexity of training from $O(TM^2NG)$ to $O(T\,(logM)^2NG)$ with only a limited loss in accuracy.

1 Introduction

The work described in this paper was carried out during the CLASS project[1]. The central objective of this project is to develop advanced learning methods that allow images, video and associated text to be analyzed and structured automatically. One of the goals of the project is the alignment of visual and textual information. We will, for example, learn the correspondence between faces in an image and persons described in surrounding text. The role of the authors in the CLASS project is mainly on information extraction from text.

In the first phase of the project we build a classifier for automatic identification and categorization of entities in texts which we report here. This classifier extracts entities from text, and assigns a label to these entities chosen from an inventory of possible labels. This task is closely related to both named entity recognition (NER), which traditionally assigns nouns to a small number of categories and word sense disambiguation (Agirre and Rigau, 1996; Yarowsky, 1995), where the sense for a word is chosen from a much larger inventory of word senses.

We will employ a probabilistic model that's been used successfully in NER (Conditional Random Fields) and use this with an extensive inventory of word senses (the WordNet lexical database) to perform entity detection.

In section 2 we describe WordNet and it's use for entity categorization. Section 3 gives an overview of Conditional Random Fields and section 4 explains how the parameters of this model are estimated during training. We will drastically reduce the computational complexity of training in section 5. Section 6 describes the implementation of this method, section 7 the obtained results and finally section 8 future work.

2 WordNet

WordNet (Fellbaum et al., 1998) is a lexical database whose design is inspired by psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized in synsets. A synset is a collection of words that have a close meaning and that represent an underlying concept. An example of such a synset is "person, individual, someone, somebody, mortal, soul". All these words refer to a human being.

WordNet (v2.1) contains 155.327 words, which are organized in 117.597 synsets. WordNet defines a number of relations between synsets. For nouns the most important relation is the hypernym/hyponym relation. A noun X is a hypernym of a noun Y if Y is a subtype or instance of X. For example, "bird" is a hypernym of "penguin" (and "penguin" is a hyponym of "bird"). This relation organizes the synsets in a hierarchical tree (Hayes, 1999), of which a fragment is pictured in fig. 1.

[1] http://class.inrialpes.fr/

Proceedings of the 2nd Workshop on Ontology Learning and Population, pages 33–40,
Sydney, July 2006.

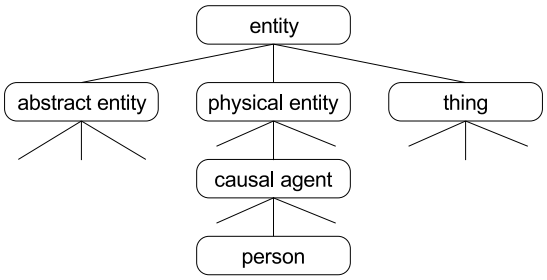

Figure 1: Fragment of the hypernym/hyponym tree

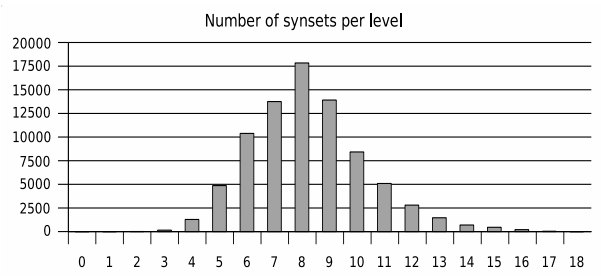

Figure 2: Number of synsets per level in WordNet

This tree has a depth of 18 levels and maximum width of 17837 synsets (fig. 2).

We will build a classifier using CRFs that tags noun phrases in a text with their WordNet synset. This will enable us to recognize entities, and to classify the entities in certain groups. Moreover, it allows learning the context pattern of a certain meaning of a word. Take for example the sentence "The ambulance took the remains of the bomber to the morgue." Having every noun phrase tagged with it's WordNet synset reveals that in this sentence, "bomber" is "a person who plants bombs" (and not "a military aircraft that drops bombs during flight"). Using the hypernym/hyponym relations from WordNet, we can also easily find out that "ambulance" is a kind of "car", which in turn is a kind of "conveyance, transport" which in turn is a "physical object".

3 Conditional Random Fields

Conditional random fields (CRFs) (Lafferty et al., 2001; Jordan, 1999; Wallach, 2004) is a statistical method based on undirected graphical models. Let X be a random variable over data sequences to be labeled and Y a random variable over corresponding label sequences. All components Y_i of Y are assumed to range over a finite label alphabet K. In this paper X will range over the sentences of a text, tagged with POS-labels and Y ranges over the synsets to be recognized in these sentences.

We define $G = (V, E)$ to be an undirected graph such that there is a node $v \in V$ corresponding to each of the random variables representing an element Y_v of Y. If each random variable Y_v obeys the Markov property with respect to G (e.g., in a first order model the transition probability depends only on the neighboring state), then the model (Y, X) is a Conditional Random Field. Although the structure of the graph G may be arbitrary, we limit the discussion here to graph structures in which the nodes corresponding to elements of Y form a simple first-order Markov chain.

A CRF defines a conditional probability distribution $p(Y|X)$ of label sequences given input sequences. We assume that the random variable sequences X and Y have the same length and use $\mathbf{x} = (x_1, ..., x_T)$ and $\mathbf{y} = (y_1, ..., y_T)$ for an input sequence and label sequence respectively. Instead of defining a joint distribution over both label and observation sequences, the model defines a conditional probability over labeled sequences. A novel observation sequence $\mathbf{x}$ is labeled with $\mathbf{y}$, so that the conditional probability $p(\mathbf{y}|\mathbf{x})$ is maximized. We define a set of K binary-valued features or feature functions $f_k(y_{t-1}, y_t, \mathbf{x})$ that each express some characteristic of the empirical distribution of the training data that should also hold in the model distribution. An example of such a feature is

$$f_k(y_{t-1}, y_t, \mathbf{x}) = \begin{cases} 1 & \text{if } x \text{ has POS 'NN' and} \\ & y_t \text{ is concept 'entity'} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Feature functions can depend on the previous (y_{t-1}) and the current (y_t) state. Considering K feature functions, the conditional probability distribution defined by the CRF is

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} exp \left\{ \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}) \right\} \quad (2)$$

where λ_j is a parameter to model the observed statistics and $Z(\mathbf{x})$ is a normalizing constant computed as

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in Y} exp \left\{ \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}) \right\}$$

This method can be thought of a generalization of both the Maximum Entropy Markov model (MEMM) and the Hidden Markov model (HMM).

It brings together the best of discriminative models and generative models: (1) It can accommodate many statistically correlated features of the inputs, contrasting with generative models, which often require conditional independent assumptions in order to make the computations tractable and (2) it has the possibility of context-dependent learning by trading off decisions at different sequence positions to obtain a global optimal labeling. Because CRFs adhere to the maximum entropy principle, they offer a valid solution when learning from incomplete information. Given that in information extraction tasks, we often lack an annotated training set that covers all possible extraction patterns, this is a valuable asset.

Lafferty et al. (Lafferty et al., 2001) have shown that CRFs outperform both MEMM and HMM on synthetic data and on a part-of-speech tagging task. Furthermore, CRFs have been used successfully in information extraction (Peng and McCallum, 2004), named entity recognition (Li and McCallum, 2003; McCallum and Li, 2003) and sentence parsing (Sha and Pereira, 2003).

4 Parameter estimation

In this section we'll explain to some detail how to derive the parameters $\theta = \{\lambda_k\}$, given the training data. The problem can be considered as a constrained optimization problem, where we have to find a set of parameters which maximizes the log likelihood of the conditional distribution (McCallum, 2003). We are confronted with the problem of efficiently calculating the expectation of each feature function with respect to the CRF model distribution for every observation sequence x in the training data. Formally, we are given a set of training examples $D = \left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i=1}^{N}$ where each $\mathbf{x}^{(i)} = \left\{x_1^{(i)}, x_2^{(i)}, ..., x_T^{(i)}\right\}$ is a sequence of inputs and $\mathbf{y}^{(i)} = \left\{y_1^{(i)}, y_2^{(i)}, ..., y_T^{(i)}\right\}$ is a sequence of the desired labels. We will estimate the parameters by penalized maximum likelihood, optimizing the function:

$$l(\theta) = \sum_{i=1}^{N} log\, p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \qquad (3)$$

After substituting the CRF model (2) in the likelihood (3), we get the following expression:

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_{t-1}^{(i)}, y_t^{(i)}, \mathbf{x}^{(i)}) \\ &\quad - \sum_{i=1}^{N} log\, Z(\mathbf{x}^{(i)}) \end{aligned}$$

The function $l(\theta)$ cannot be maximized in closed form, so numerical optimization is used. The partial derivates are:

$$\begin{aligned} \frac{\partial l(\theta)}{\partial \lambda_k} &= \sum_{i=1}^{N}\sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}^{(i)}) \\ &\quad - \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{y,y'} f_k(y', y, \mathbf{x}^{(i)})\, p(y', y|\mathbf{x}^{(i)}) \end{aligned} \qquad (4)$$

Using these derivates, we can iteratively adjust the parameters θ (with Limited-Memory BFGS (Byrd et al., 1994)) until $l(\theta)$ has reached an optimum. During each iteration we have to calculate $p(y', y|x^{(i)})$. This can be done, as for the Hidden Markov Model, using the forward-backward algorithm (Baum and Petrie, 1966; Forney, 1996). This algorithm has a computational complexity of $O(TM^2)$ (where T is the length of the sequence and M the number of the labels). We have to execute the forward-backward algorithm once for every training instance during every iteration. The total cost of training a linear-chained CRFs is thus:

$$O(TM^2NG)$$

where N is the number of training examples and G the number of iterations. We've experienced that this complexity is an important delimiting factor when learning a big collection of labels. Employing CRFs to learn the 95076 WordNet synsets with 20133 training examples was not feasible on current hardware. In the next section we'll describe the method we've implemented to drastically reduce this complexity.

5 Reducing complexity

In this section we'll see how we create groups of features for every label that enable an important reduction in complexity of both labeling and training. We'll first discuss how these groups of features are created (section 5.1) and then how both labeling (section 5.2) and training (section 5.3) are performed using these groups.

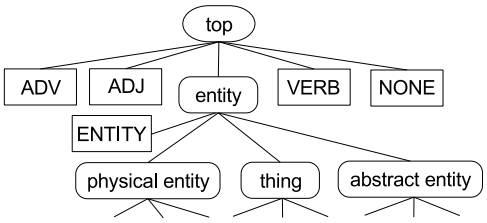

Figure 3: Fragment of the tree used for labeling

5.1 Hierarchical feature selection

To reduce the complexity of CRFs, we assign a selection of features to every node in the hierarchical tree. As discussed in section 2 WordNet defines a relation between synsets which organises the synsets in a tree. In its current form this tree does not meet our needs: we need a tree where every label used for labeling corresponds to exactly one leaf-node, and no label corresponds to a non-leaf node. We therefor modify the existing tree. We create a new top node ("top") and add the original tree as defined by WordNet as a subtree to this top-node. We add leaf-nodes corresponding to the labels "NONE", "ADJ", "ADV", "VERB" to the top-node and for the other labels (the noun synsets) we add a leaf-node to the node representing the corresponding synset. For example, we add a node corresponding to the label "ENTITY" to the node "entity". Fig. 3 pictures a fraction of this tree. Nodes corresponding to a label have an uppercase name, nodes not corresponding to a label have a lowercase name.

We use v to denote nodes of the tree. We call the top concept v^{top} and the concept v^+ the parent of v, which is the parent of v^-. We call A_v the collection of ancestors of a concept v, including v itself.

We will now show how we transform a regular CRF in a CRF that uses hierarchical feature selection. We first notice that we can rewrite eq. 2 as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} G(y_{t-1}, y_t, \mathbf{x})$$

with $G(y_{t-1}, y_t, \mathbf{x}) = exp(\sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}))$

We rewrite this equation because it will enable us to reduce the complexity of CRFs and it has the property that $p(y_t|y_{t-1}, \mathbf{x}) \approx G(y_{t-1}, y_t, \mathbf{x})$ which we will use in section 5.3.

We now define a collection of features F_v for every node v. If v is leaf-node, we define F_v as the collection of features $f_k(y_{t-1}, y_t, \mathbf{x})$ for which it is possible to find a node v_{t-1} and input $\mathbf{x}$ for which $f_k(v_{t-1}, v, \mathbf{x}) \neq 0$. If v is a non-leaf node, we define F_v as the collection of features $f_k(y_{t-1}, y_t, \mathbf{x})$ (1) which are elements of F_{v^-} for every child node v^- of v and (2) for every v_1^- and v_2^-, children of v, it is valid that for every previous label v_{t-1} and input $\mathbf{x}$ $f_k(v_{t-1}, v_1^-, \mathbf{x}) = f_k(v_{t-1}, v_2^-, \mathbf{x})$.

Informally, F_v is the collection of features which are useful to evaluate for a certain node. For the leaf-nodes, this is the collection of features that can possibly return a non-zero value. For non-leaf nodes, it's useful to evaluate features belonging to F_v when they have the same value for all the descendants of that node (which we can put to good use, see further).

We define $F'_v = F_v \setminus F_{v^+}$ where v^+ is the parent of label v. For the top node v^{top} we define $F'_{v^{top}} = F_{v^{top}}$. We also set

$$G'(y_{t-1}, y_t, \mathbf{x}) = exp\left\{ \sum_{f_k \in F'_{y_t}} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}) \right\}$$

We've now organised the collection of features in such a way that we can use the hierarchical relations defined by WordNet when determining the probability of a certain labeling $\mathbf{y}$. We first see that

$$\begin{aligned} G(y_{t-1}, y_t, \mathbf{x}) &= exp\left\{ \sum_{f_k \in F_{y_t}} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}) \right\} \\ &= G(y_{t-1}, y_t^+, x) G'(y_{t-1}, y_t, x) \\ &= \ldots \\ &= \prod_{v \in A_{y_t}} G'(y_{t-1}, v, x) \end{aligned}$$

we can now determine the probability of a labeling $\mathbf{y}$, given input $\mathbf{x}$

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \prod_{v \in A_{y_t}} G'(y_{t-1}, v, \mathbf{x}) \quad (5)$$

This formula has exactly the same result as eq. 2. Because we assigned a collection of features to every node, we can discard parts of the search space when searching for possible labelings, obtaining an important reduction in complexity. We elaborate this idea in the following sections for both labeling and training.

5.2 Labeling

The standard method to label a sentence with CRFs is by using the Viterbi algorithm (Forney, 1973; Viterbi, 1967) which has a computational complexity of $O(TM^2)$. The basic idea to reduce this computational complexity is to select the best labeling in a number of iterations. In the first iteration, we label every word in a sentence with a label chosen from the top-level labels. After choosing the best labeling, we refine our choice (choose a child label of the previous chosen label) in subsequent iterations until we arrive at a synset which has no children. In every iteration we only have to choose from a very small number of labels, thus breaking down the problem of selecting the correct label from a large number of labels in a number of smaller problems.

Formally, when labeling a sentence we find the label sequence $\mathbf{y}$ such that $\mathbf{y}$ has the maximum probability of all labelings. We will estimate the best labeling in an iterative way: we start with the best labeling $\mathbf{y}^{top-1} = \{y_1^{top-1}, ..., y_T^{top-1}\}$ choosing only from the children $y_t^{top-1}$of the top node. The probability of this labeling $\mathbf{y}^{top-1}$ is

$$p(\mathbf{y}^{top-1}|\mathbf{x}) = \frac{1}{Z'(\mathbf{x})} \prod_{t=1}^{T} G'(y_{t-1}, y_t^{top-1}, \mathbf{x})$$

where $Z'(x)$ is an appropriate normalizing constant. We now select a labeling $\mathbf{y}^{top-2}$ so that on every position t node y_t^{top-2} is a child of y_t^{top-1}. The probabilty of this labeling is (following eq. 5)

$$p(\mathbf{y}^{top-2}|\mathbf{x}) = \frac{1}{Z'(\mathbf{x})} \prod_{t=1}^{T} \prod_{v \in A_{y_t^{top-2}}} G'(y_{t-1}, v, \mathbf{x})$$

After selecting a labeling $\mathbf{y}^{top-2}$ with maximum probability, we proceed by selecting a labeling $\mathbf{y}^{top-3}$ with maximum probability etc.. We proceed using this method until we reach a labeling in which every y_t is a node which has no children and return this labeling as the final labeling.

The assumption we make here is that if a node v is selected at position t of the most probable labeling $\mathbf{y}^{top-s}$ the children v^- have a larger probability of being selected at position t in the most probable labeling $\mathbf{y}^{top-s-1}$. We reduce the number of labels we take into consideration by stating that for every concept v for which $v \neq y_t^{top-s}$, we set $G'(y_{t-1}, v_t^-, \mathbf{x}) = 0$ for every child v^- of v. This reduces the space of possible labelings drastically, reducing the computational complexity of the Viterbi algorithm. If q is the average number of children of a concept, the depth of the tree is $log_q(M)$. On every level we have to execute the Viterbi algorithm for q labels, thus resulting in a total complexity of

$$O(T\, log_q(M) q^2) \tag{6}$$

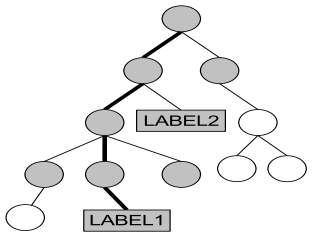

Figure 4: Nodes that need to be taken into account during the forward-backward algorithm

5.3 Training

We will now discuss how we reduce the computational complexity of training. As explained in section 4 we have to estimate the parameters λ_k that optimize the function $l(\theta)$. We will show here how we can reduce the computational complexity of the calculation of the partial derivates $\frac{\partial l(\theta)}{\partial \lambda_k}$ (eq. 4). The predominant factor with regard to the computational complexity in the evaluation of this equation is the calculation of $p(y_{t-1}, y|\mathbf{x}^{(i)})$. Recall we do this with the forward-backward algorithm, which has a computational complexity of $O(TM^2)$. We reduce the number of labels to improve performance. We will do this by making the same assumption as in the previous section: for every concept v at level s, for which $v \neq y_t^{top-s}$, we set $G'(y_{t-1}, v_t^-, \mathbf{x}) = 0$ for every child v^- of v. Since (as noted in sect. 5.2) $p(v_t|y_{t-1}, \mathbf{x}) \approx G(y_{t-1}, v_t, \mathbf{x})$, this has the consequence that $p(v_t|y_{t-1}, \mathbf{x}) = 0$ and that $p(v_t, y_{t-1}|\mathbf{x}) = 0$. Fig. 4 gives a graphical representation of this reduction of the search space. The correct label here is "LABEL1" , the grey nodes have a non-zero $p(v_t, y_{t-1}|\mathbf{x})$ and the white nodes have a zero $p(v_t, y_{t-1}|\mathbf{x})$.

In the forward backward algorithm we only have to account every node v that has a non-zero $p(v, y_{t-1}|\mathbf{x})$. As can be easily seen from fig. 4, the number of nodes is $qlog_qM$, where q is the average number of children of a concept. The total complexity of running the forward-backward algorithm is $O(T(q\, log_q M)^2)$. Since we have to run this algorithm once for every gradient compu-

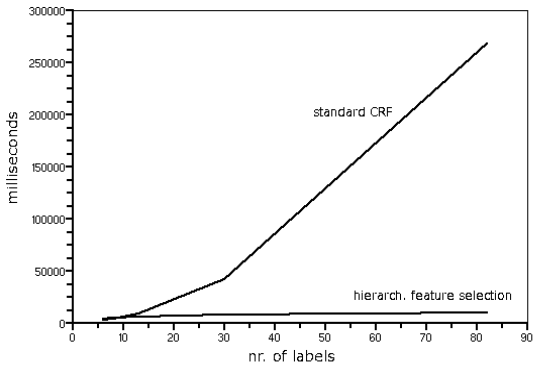

Figure 5: Time needed for one training cycle

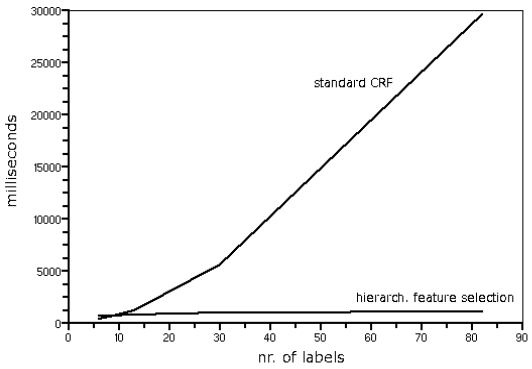

Figure 6: Time needed for labeling

tation for every training instance we find the total training cost

$$O(T(q\,log_q M)^2 NG) \tag{7}$$

6 Implementation

To implement the described method we need two components: an interface to the WordNet database and an implementation of CRFs using a hierarchical model. JWordNet is a Java interface to WordNet developed by Oliver Steele (which can be found on `http://jwn.sourceforge.net/`). We used this interface to extract the WordNet hierarchy.

An implementation of CRFs using the hierarchical model was obtained by adapting the Mallet[2] package. The Mallet package (McCallum, 2002) is an integrated collection of Java code useful for statistical natural language processing, document classification, clustering, and information extraction. It also offers an efficient implementation of CRFs. We've adapted this implementation so it creates hierarchical selections of features which are then used for training and labeling.

We used the Semcor corpus (Fellbaum et al., 1998; Landes et al., 1998) for training. This corpus, which was created by the Princeton University, is a subset of the English Brown corpus containing almost 700,000 words. Every sentence in the corpus is noun phrase chunked. The chunks are tagged by POS and both noun and verb phrases are tagged with their WordNet sense. Since we do not want to learn a classification for verb synsets, we replace the tags of the verbs with one tag "VERB".

[2]http://mallet.cs.umass.edu/

7 Results

The major goal of this paper was to build a classifier that could learn all the WordNet synsets in a reasonable amount of time. We will first discuss the improvement in time needed for training and labeling and then discuss accuracy.

We want to test the influence of the number of labels on the time needed for training. Therefor, we created different training sets, all of which had the same input (246 sentences tagged with POS labels), but a different number of labels. The first training set only had 5 labels ("ADJ", "ADV", "VERB", "entity" and "NONE"). The second had the same labels except we replaced the label "entity" with either "physical entity", "abstract entity" or "thing". We continued this procedure, replacing parent nouns labels with their children (i.e. hyponyms) for subsequent training sets. We then trained both a CRF using a hierarchical feature selection and a standard CRF on these training sets.

Fig. 5 shows the time needed for one iteration of training with different numbers of labels. We can see how the time needed for training slowly increases for the CRF using hierarchical feature selection but increases fast when using a standard CRF. This is conform to eq. 7.

Fig. 6 shows the average time needed for labeling a sentence. Here again the time increases slowly for a CRF using hierarchical feature selection, but increases fast for a standard CRF, conform to eq. 6.

Finally, fig 7 shows the error rate (on the training data) after each training cycle. We see that a standard CRF and a CRF using hierarchical feature selection perform comparable. Note that fig 7 gives the error rate on the training data but this

can differ considerable from the error rate on unseen data.

After these tests on a small section of the Semcor corpus, we trained a CRF using hierarchical feature selection on 7/8 of the full corpus. We trained for 23 iterations, which took approximately 102 hours. Testing the model on the remaining 1/8 of the corpus resulted in an accuracy of 77.82%. As reported in (McCarthy et al., 2004), a baseline approach that ignors context but simply assigns the most likely sense to a given word obtains a accuracy of 67%. We did not have the possibility to compare the accuracy of this model with a standard CRF, since as already stated, training such a CRF takes impractically long, but we can compare our systems with existing WSD-systems. Mihalcea and Moldovan (Mihalcea and Moldovan, 1999) use the semantic density between words to determine the word sense. They achieve an accuracy of 86.5% (testing on the first two tagged files of the Semcor corpus). Wilks and Stevenson (Wilks and Stevenson, 1998) use a combination of knowledge sources and achieve an accuracy of 92%[3]. Note that both these methods use additional knowledge apart from the WordNet hierarchy.

The sentences in the training and testing sets were already (perfectly) POS-tagged and noun chunked, and that in a real-life situation additional preprocessing by a POS-tagger (such as the LT-POS-tagger[4]) and noun chunker (such as described in (Ramshaw and Marcus, 1995)) which will introduce additional errors.

8 Future work

In this section we'll discuss some of the work we plan to do in the future. First of all we wish to evaluate our algorithm on standard test sets, such as the data of the Senseval conference[5], which tests performance on word sense disambiguation, and the data of the CoNLL 2003 shared task[6], on named entity recognition.

An important weakness of our algorithm is the fact that, to label a sentence, we have to traverse the hierarchy tree and choose the correct synsets at every level. An error at a certain level can not be recovered. Therefor, we would like to perform some a of beam-search (Bisiani, 1992), keeping a number of best labelings at every level. We strongly suspect this will have a positive impact on the accuracy of our algorithm.

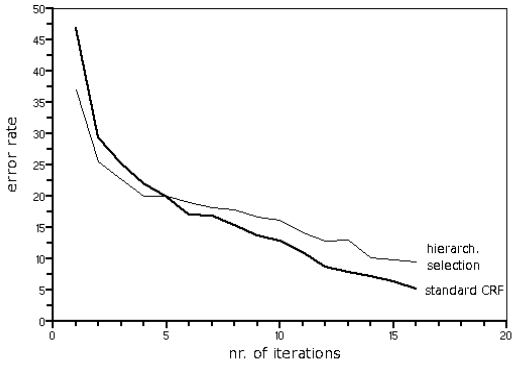

Figure 7: Error rate during training

As already mentioned, this work is carried out during the CLASS project. In the second phase of this project we will discover classes and attributes of entities in texts. To accomplish this we will not only need to label nouns with their synset, but we also need to label verbs, adjectives and adverbs. This can become problematic as WordNet has no hypernym/hyponym relation (or equivalent) for the synsets of adjectives and adverbs. WordNet has an equivalent relation for verbs (hypernym/troponym), but this structures the verb synsets in a big number of loosely structured trees, which is less suitable for the described method. VerbNet (Kipper et al., 2000) seems a more promising resource to use when classifying verbs, and we will also investigate the use of other lexical databases, such as ThoughtTreasure (Mueller, 1998), Cyc (Lenat, 1995), Openmind Commonsense (Stork, 1999) and FrameNet (Baker et al., 1998).

Acknowledgments

The work reported in this paper was supported by the EU-IST project CLASS (Cognitive-Level Annotation using Latent Statistical Structure, IST-027978).

References

Eneko Agirre and German Rigau. 1996. Word sense disambiguation using conceptual density. In *Proceedings of the 16th International Conference on*

[3]This method was tested on the Semcore corpus, but use the word senses of the Longman Dictionary of Contemporary English

[4]http://www.ltg.ed.ac.uk/software/

[5]http://www.senseval.org/

[6]http://www.cnts.ua.ac.be/conll2003/

Computational Linguistics (Coling'96), pages 16–22, Copenhagen, Denmark.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The Berkeley Framenet project. In *Proceedings of the COLING-ACL*.

L. E. Baum and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics,*, 37:1554–1563.

R. Bisiani. 1992. Beam search. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, New York. Wiley-Interscience.

Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. 1994. Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.*, 63(2):129–156.

C. Fellbaum, J. Grabowski, and S. Landes. 1998. Performance and confidence in a semantic annotation task. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press.

G. D. Forney. 1973. The viterbi algorithm. In *Proceeding of the IEEE*, pages 268 – 278.

G. D. Forney. 1996. The forward-backward algorithm. In *Proceedings of the 34th Allerton Conference on Communications, Control and Computing*, pages 432–446.

Brian Hayes. 1999. The web of words. *American Scientist*, 87(2):108–112, March-April.

Michael I. Jordan, editor. 1999. *Learning in Graphical Models*. The MIT Press, Cambridge.

K. Kipper, H.T. Dang, and M. Palmer. 2000. Class-based construction of a verb lexicon. *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-2000)*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

S. Landes, C. Leacock, and R.I. Tengi. 1998. Building semantic concordances. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press.

D. B. Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):32–38.

Wei Li and Andrew McCallum. 2003. Rapid development of hindi named entity recognition using conditional random fields and feature induction. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):290–294.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 188–191. Edmonton, Canada.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

A. McCallum. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. Using automatically acquired predominant senses for word sense disambiguation. In *Proceedings of the ACL SENSEVAL-3 workshop*, pages 151–154, Barcelona, Spain.

R. Mihalcea and D.I. Moldovan. 1999. A method for word sense disambiguation of unrestricted text. In *Proceedings of the 37th conference on Association for Computational Linguistics*, pages 152–158. Association for Computational Linguistics Morristown, NJ, USA.

Erik T. Mueller. 1998. *Natural language processing with ThoughtTreasure*. Signiform, New York.

F. Peng and A. McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 329–336.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge MA, USA.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, HLT-NAACL*.

D. Stork. 1999. The openmind initiative. *IEEE Intelligent Systems & their applications*, 14(3):19–20.

A. J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theory*, 13:260–269.

Hanna M. Wallach. 2004. Conditional random fields: An introduction. Technical Report MS-CIS-04-21., University of Pennsylvania CIS.

Y. Wilks and M. Stevenson. 1998. Word sense disambiguation using optimised combinations of knowledge sources. *Proceedings of COLING/ACL*, 98.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.