Active Learning for Dependency Parsing Using Partially Annotated Sentences

Seyed Abolghasem Mirroshandel **Alexis Nasr**
Laboratoire d'Informatique Fondamentale de Marseille- CNRS - UMR 6166
Université Aix-Marseille, Marseille, France
ghasem.mirroshandel@lif.univ-mrs.fr, alexis.nasr@lif.univ-mrs.fr

Abstract

Current successful probabilistic parsers require large treebanks which are difficult, time consuming, and expensive to produce. Some parts of these data do not contain any useful information for training a parser. Active learning strategies allow to select the most informative samples for annotation. Most existing active learning strategies for parsing rely on selecting uncertain sentences for annotation. We show in this paper that selecting full sentences is not an optimal solution and propose a way to select only subparts of sentences.

1 Introduction

Current probabilistic parsers rely on treebanks in order to estimate their parameters. Such data is known to be difficult and expensive to produce. One can also observe that the quality of parsers increase when they model complex lexico-syntactic phenomena. Unfortunately, increasing the precision of models is quickly confronted with data sparseness which prevents to reliably estimate the model parameters. Treebanks size is therefore a limit to the accuracy of probabilistic parsers. Given the cost of annotating new data with syntactic structures, it is natural to ask oneself, before annotating a new sentence and adding it to a treebank, if this new piece of information will benefit a parser trained on the treebank.

This question is at the heart of active learning which assumes that *"a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g. human annotator). Active learning is well-motivated in many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain."* (Settles, 2010)

Different scenarios have been proposed for active learning. The work described here is based on the Pool-Based Sampling scenario (Lewis and Gale, 1994) which is adapted to the problem at hand. Pool-based sampling scenario assumes that there is a small set of labeled data $\mathcal{L}$ and a large pool of unlabeled data $\mathcal{U}$ available. Instances are drawn from $\mathcal{U}$ according to a selection strategy. The chosen instance(s) are then labeled and added to $\mathcal{L}$. These steps are repeated until either $\mathcal{U}$ is empty or does not contain informative instances anymore.

Many selection strategies have been proposed in the literature. Uncertainty sampling, which is the strategy that has been used in this work, is the simplest and the most commonly used one. In this strategy, instances for which the prediction of the label is the most uncertain are selected by the learner. The performances of such methods heavily rely on the definition of a good confidence measure, which measures the confidence the model, in our case the parser, has in the solution it proposes.

Active learning has been used for many NLP applications, such as automatic speech recognition, information extraction, part of speech tagging or syntactic parsing. We will not detail the way active learning has been applied to those tasks, the interested reader can find two reasonably recent surveys in (Settles, 2010) and (Olsson, 2009), and concentrate on its application to statistical parsing.

(Tang et al., 2002) and (Hwa, 2004) proposed active learning techniques for training probabilistic parsers. They both used uncertainty sampling and suggested a measure of uncertainty based on the entropy of the probability distribution of the parses of a sentence. The idea being that the higher this entropy is, the more uncertain the

Proceedings of the 12th International Conference on Parsing Technologies, pages 140–149,
October 5-7, 2011, Dublin City University.

parser is of its choice. (Tang et al., 2002) proposed to combine uncertainty with representativeness, the idea here is to make sure that the instances selected with uncertainty criterion are also representative of the data distribution. The common point to these two approaches, and most work on active learning applied to parsing is that they only consider full sentences in their instance selection.

There is something not satisfactory in considering only full sentences as instances since, in many cases, only part of the syntactic structure is uncertain. The manual annotation of this part would benefit the parser while the annotation of the remaining part of the sentence would be a waste of effort.

The main novelty of the work reported here is that we explore the other extreme position: instead of considering uncertain sentences, we consider single uncertain word tokens in a sentence. More precisely, we consider the attachment of single word tokens. In the framework of dependency parsing, which is used here, this boils down to selecting, for a token w, its governor and the label of the dependency between w and its governor. The task is therefore to select, in a sentence, the most uncertain dependencies.

Active learning for parsing using sub-sentential units has already been explored by (Sassano and Kurohashi, 2010). In their work they select unreliable dependencies predicted by a parser (a simple shift-reduce parser where dependencies are weighted using an averaged perceptron with polynomial kernels), based on the score the parser assigns to the dependencies. Such dependencies are hand labeled and, based on syntactic characteristics of Japanese syntax (dependencies are projective and oriented from left to right (the governor is always to the right of the dependent)) other dependencies are deduced and this set of dependencies are added to the training set. Our work departs from their in the uncertainty measure that we use, in the kind of parser used, on the way hand annotated single dependencies are added to the training data (without using language specific properties) as well as on the fact that we use both sentence based and dependency based uncertainty measures.

The structure of the paper is the following: in section 2, we describe, in some details, the parsing model on which this work is based, namely the graph-based dependency parsing framework. Such a detailed description is motivated by two extensions of graph based parsing that we describe. The first one is how to take into account, in the input of the parser, part of the syntactic structure of the sentence to parse, a feature that is needed when annotating only a subpart of the syntactic structure of a sentence. We will see that this can be done very easily and this is the main reason why we chose this parsing framework. The second extension concerns the production of a list of n-best candidates, which is necessary for computing our uncertainty measure and is not a standard operation for this kind of parsers, contrary to syntagmatic parsers, for example. In section 3, we report a first series of experiments in which full sentences are selected during the instance selection procedure. The techniques used here are not novel and the aim of this section is to constitute a baseline for evaluating our idea of partial annotation, which is described in section 4. Section 5 shows that better results can be reached when taking into account both uncertainty of a sentence and uncertainty of parts of it. Finally, section 6 concludes the paper.

2 Graph-Based Parsing

Graph-Based parsing (McDonald et al., 2005; Kübler et al., 2009) defines a framework for parsing that does not make use of a generative grammar. In such a framework, given a sentence $W = w_1 \dots w_l$, any dependency tree[1] for W is a valid syntactic representation of W. The heart of a graph-based parser is the scoring function that assigns a score to every tree $T \in \mathcal{T}_W$, where $\mathcal{T}_W$ denotes the set of all dependency trees of sentence W. Such scores are usually a weighted sum of the scores of subparts of W.

$$s(T) = \sum_{\psi \in \psi_T} \lambda_\psi$$

where ψ_T is the set of all the relevant subparts of tree T and λ_ψ is the score of subpart ψ.

Depending on the decomposition of the tree into subparts, different models of increasing complexity can be defined. The most simple one is the *arc factored model*, also called *first order model*, which simply decomposes a tree into single

[1] A dependency tree for sentence $W = w_1 \dots w_l$ and the dependency relation set $\mathcal{R}$ is a directed labelled tree (V, A) such that $V = \{w_1, \dots w_n\}$ and $A \subseteq V \times \mathcal{R} \times V$.

dependencies and assigns a score to a dependency, independently of its context.

The score of tree T, noted $s(T)$ in this model is therefore :

$$s(T) = \sum_{(w_i,r,w_j)} \lambda(i,r,j)$$

where (i,r,j) is the dependency that has w_i as governor, w_j as dependent and r as label, and $\lambda(i,r,j)$ is the score assigned to such a dependency.

Parsing a sentence W in such a system boils down to determining the tree that has the highest score in $\mathcal{T}_W$. Determining such a tree can be simply accomplished using the maximum spanning tree algorithm.

A maximum spanning tree or MST of a directed multigraph $G = (V, A)$ is the highest scoring subgraph G' that satisfies the following conditions:

- $V' = V$, i.e. G' spans all the original nodes of G.
- G' is a directed tree.

Given a sentence $W = w_1 \dots w_l$, and a set of functional labels $\mathcal{R} = \{r_1, \dots r_m\}$, a graph $G_W = (V_W, A_W)$ can be built such that:

- $V_W = \{w_1, \dots w_l\}$
- $A_W = \{(i,r,j)\}\ \forall w_i, w_j \in V_W$ and $r \in \mathcal{R}$ and $i \neq j$

In other words, G_W is the graph composed of all the dependencies that can be defined between tokens of W. Given a scoring function λ that assigns a score to every arc of G_W, a simple Graph-Based parser can be implemented based on any MST algorithm such as Chu-Liu-Edmonds (Chu and Liu, 1965; Edmonds et al., 1968).

In order to build an actual parser, a scoring function λ is needed. Current versions of Graph-Based Parsers assume that such a function is a linear classifier. The score of a dependency is computed as follows:

$$\lambda(i,r,j) = \alpha \cdot f(i,r,j)$$

where f is a feature function that maps the dependency to a boolean vector and α is the feature weight vector that associates a weight with every feature of the model. In first order models, features describe different aspects of a dependency, such as the part of speech of the governor and the dependent, their lexical value, the length of the dependency, the part of speech of tokens between the dependent and the governor ...

Different Machine Learning techniques can be used in order to learn the weight vector. In our experiment, we used the implementation of (Bohnet, 2010) that is based on MIRA (Crammer et al., 2006), an on-line large margin classifier.

The arc factored model, described here, achieves good performances, but it relies on a very strong independence assumption which is that the score of a dependency is independent of its context. Such an independence assumption is linguistically unappealing and more elaborate models, known as second order (Carreras, 2007) and third order (Koo and Collins, 2010) models, which associate a score with pairs of adjacent dependencies in a tree or chains of two or three dependencies, have shown to give better results.

Although we have used second order models in our experiments, we will consider in the next two subsections, only first order models, which are conceptually and computationally simpler.

2.1 Constraining the Output of the Parser

The graph-based framework allows us to impose structural constraints on the parses that are produced by the parser in a straightforward way, a feature that will reveal itself to be precious for our experiments.

Given a sentence $W = w_1 \dots w_l$, a parser with a scoring function λ and a dependency $d = (i,r,j)$ with $1 \leq i,j \leq l$. We can define a new scoring function λ_d^+ in the following way:

$$\lambda_d^+(i',r',j') = \begin{cases} -\infty & \text{if } j' = j \text{ and} \\ & (i' \neq i \text{ or } r' \neq r) \\ \lambda(i',r',j') & \text{otherwise} \end{cases}$$

When running the parser on sentence W, with the scoring function λ_d^+, one can be sure that dependency d will be part of the solution produced by the parser since it will be given a better score than any competing dependency, i.e. a dependency of the form (i',r,j') with $j = j'$ and $i' \neq i$ or $r' \neq r$. This feature will be used in section 4

in order to parse a sentence for which a certain number of dependencies are set in advance.

Symmetrically, we can define a function λ_d^- in the following way:

$$\lambda_d^-(i', r', j') = \begin{cases} -\infty & \text{if } j' = j \text{ and} \\ & i' = i \text{ and } r' = r \\ \lambda(i, r, j) & \text{otherwise} \end{cases}$$

The effect of this new scoring function when parsing sentence W is that the solution produced by the parser will not contain dependency d.

Functions λ^+ and λ^- can be extended in order to force the presence or the absence of any set of dependencies in the output of the parser. Suppose $\mathcal{D}$ is a set of dependencies, we define $\lambda_{\mathcal{D}}^+$ the following way:

$$\lambda_{\mathcal{D}}^+(i, r, j) = \begin{cases} \lambda_d^+(i, r, j) & \text{if } (\cdot, \cdot, j) \in \mathcal{D} \\ \\ \lambda(i, r, j) & \text{otherwise} \end{cases}$$

2.2 Producing N-Best Parses

Given scoring functions $\lambda_{\mathcal{D}}^-$, we can define a simple way of producing the n-best scoring parses of a sentence. Given a scoring function λ and a sentence W, let's call $\hat{T}_1 = \{d_1, \dots d_l\}$ the tree output by the parser, where $d_1 \dots d_l$ are the dependencies that make up the tree $\hat{T}_1$, d_i being the dependency that has w_i as a dependent.

By definition, $\hat{T}_1$ is the tree of $\mathcal{T}_W$ with the highest score. Now let's define l functions $\lambda_{d_1}^- \dots \lambda_{d_l}^-$ and run the parser l times on W, equipped respectively with functions $\lambda_{d_1}^- \dots \lambda_{d_l}^-$. The result of these runs is a set of trees $\mathcal{L}_0 = \{T_1, \dots T_l\}$ respectively produced with functions $\lambda_{d_1}^- \dots \lambda_{d_l}^-$. These trees are such that T_i does not contain dependency d_i. Let's define $\hat{T}_2$ as follows:

$$\hat{T}_2 = \arg\max_{T \in \mathcal{L}_0} s(T)$$

It is easy to prove that $\hat{T}_2$ is the second best scoring tree in $\mathcal{T}_W$. This proposition holds because trees $T_1 \dots T_l$ are all the best scoring trees of $\mathcal{T}_W$ that differ from $\hat{T}_1$ by at least one dependency. Among them, the tree with the highest score is therefore the second best scoring tree.

Suppose $\hat{T}_2 = \{d'_1, \dots d'_n\}$, The process is recursively applied to $\hat{T}_2$ and produces a list of trees $\mathcal{L}$. A new list $\mathcal{L}_1$ is then defined as follows:

$$\mathcal{L}_1 = \mathcal{L}_0 - \{\hat{T}_2\} \cup \mathcal{L}$$

and the third best scoring tree $\hat{T}_3$ is defined as follows:

$$\hat{T}_3 = \arg\max_{T \in \mathcal{L}_1} s(T)$$

The process is iterated until a tree $\hat{T}_n$ has been produced, for a given value of n.

The process, as described is not computationnaly optimal since the parser is run $n \times l$ times on the same sentence and many operations are duplicated. Much of the redundant work could be avoided using dynamic programming.

We are not aware of a general method for efficiently computing n-best parses for graph-based parsing, as it is the case with context-free parsing (Huang and Chiang, 2005). Nevertheless, (Hall, 2007) describes an efficient n-best method for graph-based parsing which is unfortunately limited to first order models.

3 Selecting Full Sentences

We report in this section a first series of experiments in which full sentences are selected from the pool at every iteration of the active learning algorithm. The aim of this section is primarily to set up a baseline against which we will compare the results obtained in sections 4 and 5. In order to set up a reasonable baseline, we used successful methods described in (Tang et al., 2002) and (Hwa, 2004) but we adapted them to our framework: discriminative dependency parsing. The method described here will also prove to be interesting to combine with the model of section 4, as we will see in section 5.

The experiments have been conducted on the French Treebank (Abeillé et al., 2003) which is made of $12,350$ sentences taken from newspaper *Le Monde* and annotated with syntagmatic structures along with syntactic functions. The corpus has been divided into three parts, the labeled set $\mathcal{L}$ made of 500 sentences, the pool $\mathcal{U}$, made of $10,665$ sentences and a test set $\mathcal{T}$ made of $1,185$ sentences. The syntagmatic annotations have been converted to dependencies using the BONSAÏ converter (Candito et al., 2010).

As described in the introduction, the algorithm is Pool-Based. A first parser P_0 is trained on $\mathcal{L}$ and used to parse the sentences of $\mathcal{U}$. An uncertainty

measure is then computed for every sentence of $\mathcal{U}$ and the k most uncertain sentences are labeled, removed from $\mathcal{U}$ and added to $\mathcal{L}$. A second parser P_1 is trained on $\mathcal{L}$ and the process is iterated until there is no more sentences left in $\mathcal{U}$. These different steps are illustrated in Algorithm 1.

Algorithm 1 ACTIVE LEARNING WITH FULL SENTENCES ANNOTATION

$\mathcal{L}$: Initial training set
$\mathcal{U}$: Unlabeled pool
$\varphi(S)$: Uncertainty measure of sentence S
$i = 0$;
while $\mathcal{U}$ is not empty **do**
  $P_i = train(\mathcal{L})$;
  Build n-best parses of $\mathcal{U}$ based on P_i;
  Compute $\varphi(S)$ for $S \in \mathcal{U}$;
  $\mathcal{U}'$ = k most uncertain sentences of $\mathcal{U}$;
  $\mathcal{L}'$ = Annotate $\mathcal{U}'$;
  $\mathcal{U} = \mathcal{U} - \mathcal{U}'$;
  $\mathcal{L} = \mathcal{L} \cup \mathcal{L}'$;
  $i = i + 1$;
end while

3.1 Sentence Entropy

The uncertainty measure we have been using, the sentence entropy, noted SE, is close to the uncertainty measures defined in (Tang et al., 2002), (Hwa, 2004) or (Sánchez-Sáez et al., 2009). Given a sentence W and the n-best parses of W with scores respectively noted $s_1 \dots s_n$, $SE(W)$ is computed as follows:

$$SE(W) = -\sum_{i=1}^{n} p_i \log(p_i)$$

where p_i is the posterior probability:

$$p_i = \frac{s_i}{\sum_{j=1}^{n} s_j}$$

This uncertainty measure is based on the intuitive idea that when the scores of the n-best parses of a sentence are close to each other, and therefore the entropy of the distribution of the posterior probabilities is high, this indicates that the parser is "hesitating" between some of the n-best parses.

Two curves have been represented in figure 1, they show the Labeled Accuracy Score (LAS) computed on the test set against the size of the train set, given in tokens[2]. Two curves have been drawn, a curve based on the sentence entropy and a curve based on a random selection. The sentence entropy curve has two interesting features, it grows quicker than the random curve and presents an asymptotical shape which shows that it has "extracted" all the useful information present in $\mathcal{U}$ when it reaches the asymptote.

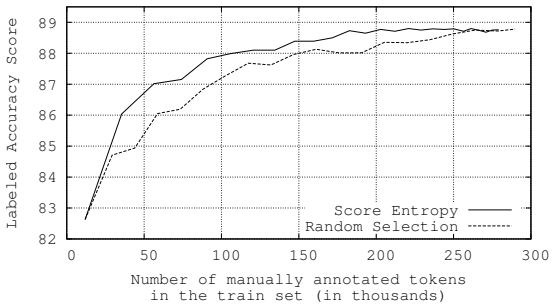

Figure 1: Learning curves with full sentences selection using sentence entropy, comparison with random selection.

The uncertainty measure of a sentence W used in this experiment is supposed to measure the difficulty of parsing sentence W. It must therefore be related to the mean error rate of the parser on sentence W. The curve of figure 2 shows the relation between labeled accuracy score and sentence entropy computed on a set of 1000 randomly selected sentences. The curve shows that on average LAS tends to decrease when sentence entropy increases. The small range in which score entropy takes its values is explained by the fact that, on average, the score of parse trees in n-best list are close to each other. Their posterior probability is therefore close to 0.01 (due to the fact that it is computed on a 100 best list). We do not have an explanation for the general shape of the curve.

(Tang et al., 2002) and (Hwa, 2004) discuss the relation between entropy and the length of sentences. The idea is that long sentences tend to have more parses than short sentences and tend to have on average higher entropy. This dependency between sentence length and entropy

[2]In all our experiments, the LAS was computed on whole sentences, including punctuation.

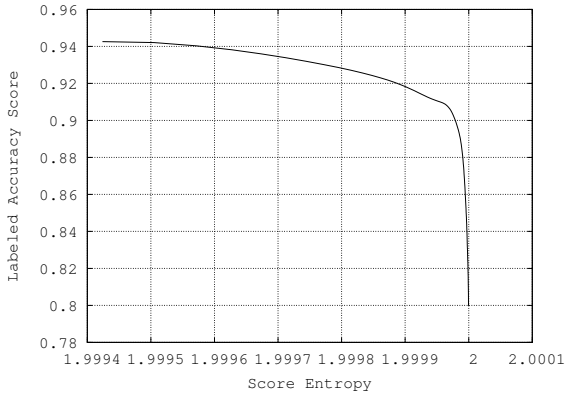

Figure 2: Relation between Labeled Accuracy Score and Sentence Entropy

will therefore bias the active learning algorithm towards selecting long sentences. They therefore propose to normalize entropy either with sentence length (Tang et al., 2002) or number of parses for a sentence (Hwa, 2004).

In our case, we did not observe any influence of sentence length on sentence entropy. That might be due to the fact that we only consider the n-best scoring parses (but that was also the case of (Tang et al., 2002)) or to the fact that the scores from which posterior probabilities are computed are not probabilities but scores given by the MIRA classifier which have different distributions than generative probabilites. We did not investigate further this difference since our goal in this section was mostly to define a reasonable baseline for evaluating the results of the selection strategy that will be described in the following section.

4 Selecting Single Tokens

The algorithm described in the preceding section associates an uncertainty measure with whole sentences and, based on this measure, decides to ask for its labeling, or not. The syntactic analysis of a sentence is a complex object, part of it can be difficult to parse and a correct analysis of this part could improve the parser while the remaining of the syntactic structure might not contain any new piece of information. The idea that we explore in this section consists in locating in a sentence some difficult parts and ask for the labeling of these parts only. More precisely, we will try to locate single difficult dependencies and ask for the labeling of just these dependencies. The partly labeled sentence is then parsed in such a way that the parser preserves the manually annotated dependencies. This is done by changing the scoring function of the parser, as described in 2.1. The output of the parser is then added to the set of labeled sentences $\mathcal{L}$.

In order to estimate the potential benefit of such a method, we conducted the following experiment: the parser P_0, trained on the 500 sentences of $\mathcal{L}$, was used to parse 1000 randomly selected sentences from $\mathcal{U}$. We will refer to this set of sentences as $\mathcal{U}_{1000}$. Among the $24,000$ dependencies created by the parser, $3,144$ were incorrect (governor or/and dependency label was incorrect for a given token). These dependencies were corrected (the right governor along with the right dependency label was restored) and the partially parsed sentences were parsed with P_0 preserving manual annotation. The output of this process was added to $\mathcal{L}$ and used to train a new parser $P_{partial}$. Eventually a third parser $P_{complete}$ was trained on a fully correct version of $\mathcal{U}_{1000}$ plus $\mathcal{L}$. The two parsers were evaluated on the test set $\mathcal{T}$. The result of these experiments are reported in table 1.

Model	LAS	UAS
$P_{complete}$	85.60	87.84
$P_{partial}$	85.71	87.91

Table 1: Performances of parsers trained with respectively a set of 1000 sentences fully annotated and the same set of sentences in which only wrong dependencies were hand annotated.

The figures reported in table 1 show that training a parser P on a corpus in which only the mistakes of P were corrected can lead to better results than training it on the fully annotated version of this same corpus. In our case, the partially annotated corpus contained only $3,144$ manually annotated dependencies and the fully annotated corpus contained $24,000$ ones.

If we were able to determine the set of wrong attachments made by a parser we would therefore be in a position to define a very efficient active learning algorithm. Although determining all and only the wrong attachments is clearly impossible, we can try to approximate this set using a dependency based confidence measure as described in the following subsection.

4.1 Attachment Entropy

Given the n-best parses of a sentence W, one indication of the confidence of the attachment of token w is the number of its possible governors in the n-best parses. We define attachment entropy of token w, noted $AE(w)$ as a simple measure of this confidence. Attachment entropy is computed as follows:

$$AE(w) = -\sum_{g \in G(w)} P(g, \cdot, w) \log P(g, \cdot, w)$$

where $G(w)$ is the set of the possible governors of token w in the n-best parses and $P(g, \cdot, w)$ is the probability that token w is governed by token g in the n-best parses. This probability is estimated as follows:

$$P(g, \cdot, w) = \frac{count(g, \cdot, w)}{n}$$

where $count(g, \cdot, w)$ counts the number of times token w was governed by token g in the n-best parses[3].

A perfect confidence measure should, in the experiments reported above, allow to select the wrong dependencies and only those. In order to evaluate the quality of attachment entropy, precision recall curves have been computed. For a given threshold τ of our confidence measure, the dependencies that are above τ (more precisely, the dependencies $(g, \cdot, w)$ of the first best parse such that $AE(w) \geq \tau$) are collected to form the set $\mathcal{D}_U$ of uncertain dependencies. This set is compared with the set $\mathcal{D}_I$ of incorrect dependencies (the set of $3,144$ dependencies described above). The comparison is realized with precision and recall, which are computed as follows:

$$Prec. = \frac{|\mathcal{D}_I \cap \mathcal{D}_U|}{|\mathcal{D}_U|} \quad Rec. = \frac{|\mathcal{D}_I \cap \mathcal{D}_U|}{|\mathcal{D}_I|}$$

Attachment entropy depends on the length of the n-best lists of parses produced by the parser. A high value of n will produce, on average, a higher number of possible governors for a token of a sentence and therefore, a potentially higher entropy. Three precision recall curves are reported in figure 3 for n best lists of lengths respectively

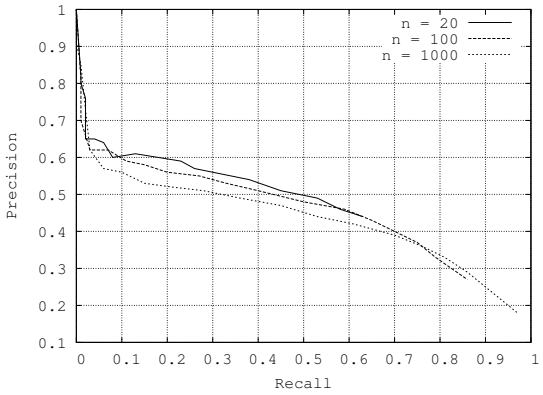

Figure 3: Precision Recall curves for attachment entropy for different n-best output of the parser.

equal to 20, 100 and 1000. Several conclusions can be drawn from these curves. The first one is that attachment entropy is quite far from the perfect confidence measure that would achieve a precision and recall of 1. High recall values can be obtained at the cost of very low precision, and inversely, high precision yields very low recall. On average, lower values of n achieve better precision recall tradeoffs but do not allow to reach high recall values.

The inability for low values of n to reach high values of recall opens the door for a discussion about the influence of the sentence length on precision recall curves. The number of parses of a sentence is, in the worse case, an exponential function of its length (Church and Patil, 1982). Computing entropy attachment on fixed length n-best parses lists only allow to consider a very limited number of possible governors of a token for longer sentences. A natural solution in order to cope with this problem is to compute variable length n-best lists, where the value of n is a function of the length (l) of the sentence being parsed. It would make sense to let n be an exponential function of l. This approach is in practice impossible since (at least in our implementation of the n best algorithm) the production of the n-best list will take an exponential time. Furthermore, our experiments showed that the value of n should not grow too fast, as reported in figure 4. Three[4] curves

[3] Labelling errors of correct attachments are not taken into account for measuring attachment entropy for they did not improve the quality of the confidence measure.

[4] We have added the fixed 20-best curve, that achieved the best result for fixed length n, for comparison purpose.

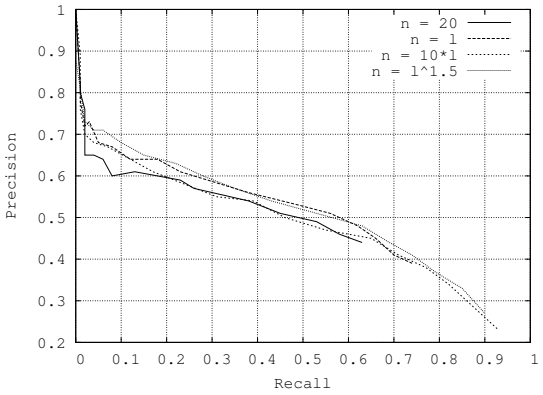

Figure 4: Precision Recall curves for attachment entropy with variable length n-best lists.

have been reported in this figure, that correspond to three functions linking n and l. The best results were obtained for the function $n = l^{\frac{3}{2}}$. Higher order function did not obtain good results, contrary to our intuition.

It is important, at this point to discuss one aspect of our attachment entropy measure. In graph-based parsing, one can compute the score of any dependency linking two tokens of a sentence. It is therefore possible to compute, for a given token w_j, the score of all the dependencies of the form $(i, \cdot, j)$, $\forall j\ 1 \leq j \leq k$ and compute attachment entropy based on this score. This method has the advantage of considering all tokens of the sentence as a potential governor and does not ask for the construction of the n-best parses. But such a method does not take into account the global score of a parse. A dependency can get a high score when considered alone but might not be part of a good (high scoring) parse. In practice, it gave very bad results[5].

4.2 Active Learning Strategy

The token based active learning algorithm is straightforward. For every iteration of the algorithm, the most uncertain tokens are selected, hand annotated and their sentences are parsed in such a way that the partial annotation is preserved. The new parses are then added to the labeled set and the process is iterated, as shown in Algorithm 2. Two different approaches can be considered when selecting most uncertain tokens. One can either select for every sentence the most uncertain token or select for the whole pool the k most uncertain tokens. In the latter case, several tokens of a single sentence could be selected for every step.

Algorithm 2 ACTIVE LEARNING WITH SINGLE DEPENDENCY ANNOTATION

$\mathcal{L}$: Initial training set
$\mathcal{U}$: Unlabeled pool
$\varphi(w)$: Uncertainty measure for token w
$i = 0$;
while There is no uncertain dependency **do**
  $P_i = train(\mathcal{L})$;
  Build n-best parses of $\mathcal{U}$ based on P_i;
  Compute $\varphi(w)$ for each token;
  $\mathcal{U}'$ = sentences containing the k most uncertain tokens;
  Annotate selected tokens;
  $\mathcal{L}'$ = Parse $\mathcal{U}'$ sentences, preserving annotation;
  $\mathcal{L} = \mathcal{L} \cup \mathcal{L}'$;
  $i = i + 1$;
end while

The result of the token selection strategy is reported in figure 5. The full sentence selection based on score entropy as well as the random selection are also reported for comparison purpose. The two strategies described above (selecting the most uncertain token per sentence or the k most uncertain tokens in the corpus) gave almost the same results. The full sentence selection strategy yielded better results for the first iterations of the algorithm but was outperformed by the token selection strategy after a while. The token selection strategy shows good asymptotical performances, it reaches the asymptote with $130,000$ manually annotated tokens (45.45% of total tokens of $\mathcal{U}$) while the full sentence strategy asks for $180,000$ tokens (62.94% of total tokens of $\mathcal{U}$) to reach it. It is unclear why full sentence selection gave better results in the first iterations, it is as if the parser needs to be trained on full sentences in the beginning and, after a while, gain better benefit from isolated difficult attachments.

[5] As suggested by one of the reviewers, for first order models, the confidence measure can be computed without producing the n-best parses, using marginal probability of an arc existing in any tree, which can be efficiently computed, via the matrix tree theorem (McDonald and Satta, 2007).

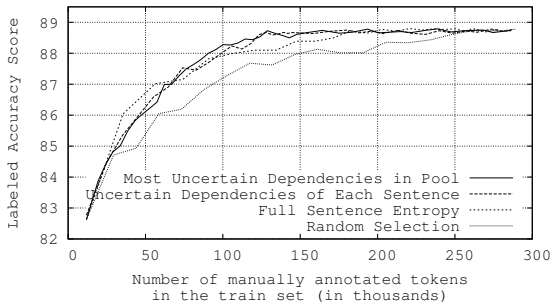

Figure 5: Learning curves with token selection strategy. Comparison with full sentence strategy and random selection.

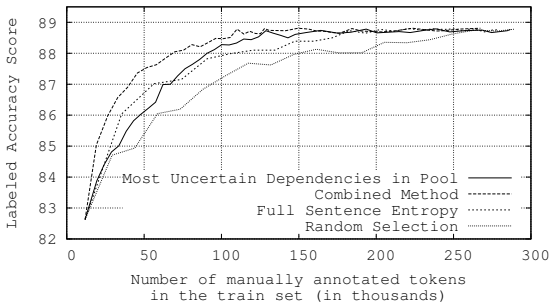

Figure 6: Learning curves for the combined strategy. Comparison with token selection strategy, full sentence strategy and random selection.

5 Selecting Uncertain Tokens of Uncertain Sentences

The results obtained when selecting uncertain sentences and those obtained for uncertain tokens showed that both strategies seem to be somehow complementary. Such conclusion opens the door for a two level strategy. In a first step uncertain full sentences are selected, then uncertain tokens of these sentences are selected, as described in Algorithm 3.

Algorithm 3 COMBINED ACTIVE LEARNING

$\mathcal{L}$: Initial training set
$\mathcal{U}$: Unlabeled pool
$\varphi(S)$: Full sentence uncertainty measure
$\psi(w)$: Single dependency uncertainty measure
$i = 0$;
while $\mathcal{U}$ is not empty **do**
  $P_i = train(\mathcal{L})$;
  Build n-best parses of $\mathcal{U}$ based on P_i;
  Compute $\varphi(S)$ for $S \in \mathcal{U}$;
  $\mathcal{U}'$ = k most uncertain sentences of $\mathcal{U}$;
  Compute $\psi(x)$ for each token of $\mathcal{U}'$;
  Select k' most uncertain tokens;
  Annotate selected tokens;
  $\mathcal{L}'$ = Partially parse $\mathcal{U}'$ sentences, preserving annotation;
  $\mathcal{U} = \mathcal{U} - \mathcal{U}'$;
  $\mathcal{L} = \mathcal{L} \cup \mathcal{L}'$;
  $i = i + 1$;
end while

Figure 6 reports the results of the combined method. The new method outperforms both the full sentence and the single token strategy for every iteration of the algorithm. It reaches the asymptote with only $109,000$ manually annotated tokens (37.98% of total tokens in $\mathcal{U}$). These result confirm the intuition that both score entropy and attachment entropy carry complementary information. One way to interpret the results obtained is that the two methods that were tested in this paper: selecting full sentences vs. selecting single tokens, are two extreme positions. We did not investigate the search space that lie between them, i.e. specifically looking for unreliable *parts* of sentences. We describe in the next section some preliminary results along this line.

6 Conclusions and Future Work

We proposed in this paper three active learning techniques for dependency parsing. The novelty of this work is to annotate only difficult attachment insead of full sentences as it is usually done in other approaches.

Although selecting single dependencies showed an improvement over selecting full sentences, in some cases, it turns out that selecting subparts of the sentence is a good strategy. A preliminary experiment was conducted on punctuation. It showed that selecting tokens that lie in a window of 6 tokens centered on punctuation yielded very good results. Such a method could be useful for difficult attachment such as PP attachment or coordination where dependencies must be considered in a larger context.

Acknowledgements

This work has been funded by the French Agence Nationale pour la Recherche, through the project SEQUOIA (ANR-08-EMER-013). The authors would like to thank the anonymous reviewers for the very high quality of their reviews.

References

Anne Abeillé, Lionel Clément, and Toussenel François, 2003. *Treebanks*, chapter Building a treebank for French. Kluwer, Dordrecht.

Bernd Bohnet. 2010. Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of COLING*.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical French Dependency Parsing: Treebank Conversion and First Results. In *Proceedings of LREC2010*.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, volume 7, pages 957–961.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14(1396-1400):270.

K. Church and R. Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8(3-4):139–149.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai ShalevShwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithm. *Journal of Machine Learning Research*.

J. Edmonds, J. Edmonds, and J. Edmonds. 1968. *Optimum branchings*. National Bureau of standards.

K. Hall. 2007. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the ACL*, page 392.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 53–64.

R. Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the ACL*, pages 1–11.

S. Kübler, R. McDonald, and J. Nivre. 2009. *Dependency parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

D.D. Lewis and W.A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, pages 121–132.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98.

F. Olsson. 2009. A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science.

R. Sánchez-Sáez, J.A. Sánchez, and J.M. Benedí. 2009. Statistical confidence measures for probabilistic parsing. In *Proceedings of RANLP*, pages 388–392.

Manabu Sassano and Sadao Kurohashi. 2010. Using smaller constituents rather than sentences in active learning for japanese dependency parsing. In *Proceedings of the 48th Annual Meeting of the ACL*, pages 356–365.

Burr Settles. 2010. Active Learning Literature Survey. Technical Report Technical Report 1648, University of Wisconsin-Madison.

M. Tang, X. Luo, and S. Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127.