Unsupervised methods for head assignments

Sangati, F.; Zuidema, W.

DOI
10.3115/1609067.1609145

Publication date
2009

Document Version
Final published version

Published in
Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics

Unsupervised Methods for Head Assignments

Federico Sangati, Willem Zuidema
Institute for Logic, Language and Computation
University of Amsterdam, the Netherlands
{f.sangati,zuidema}@uva.nl

Abstract

We present several algorithms for assigning heads in phrase structure trees, based on different linguistic intuitions on the role of heads in natural language syntax. Starting point of our approach is the observation that a head-annotated treebank defines a unique lexicalized tree substitution grammar. This allows us to go back and forth between the two representations, and define objective functions for the unsupervised learning of head assignments in terms of features of the implicit lexicalized tree grammars. We evaluate algorithms based on the match with gold standard head-annotations, and the comparative parsing accuracy of the lexicalized grammars they give rise to. On the first task, we approach the accuracy of hand-designed heuristics for English and inter-annotation-standard agreement for German. On the second task, the implied lexicalized grammars score 4% points higher on parsing accuracy than lexicalized grammars derived by commonly used heuristics.

1 Introduction

The *head* of a phrasal constituent is a central concept in most current grammatical theories and many syntax-based NLP techniques. The term is used to mark, for any nonterminal node in a syntactic tree, the specific daughter node that fulfills a special role; however, theories and applications differ widely in what that special role is supposed to be. In descriptive grammatical theories, the role of the head can range from the determinant of agreement or the locus of inflections, to the governor that selects the morphological form of its sister nodes or the constituent that is distributionally equivalent to its parent (Corbett et al., 2006).

In computational linguistics, heads mainly serve to select the lexical content on which the probability of a production should depend (Charniak, 1997; Collins, 1999). With the increased popularity of dependency parsing, head annotations have also become a crucial level of syntactic information for transforming constituency treebanks to dependency structures (Nivre et al., 2007) or richer syntactic representations (e.g., Hockenmaier and Steedman, 2007).

For the WSJ-section of the Penn Treebank, a set of heuristic rules for assigning heads has emerged from the work of (Magerman, 1995) and (Collins, 1999) that has been employed in a wide variety of studies and proven extremely useful, even in rather different applications from what the rules were originally intended for. However, the rules are specific to English and the treebank's syntactic annotation, and do not offer much insights into how headedness can be learned in principle or in practice. Moreover, the rules are heuristic and might still leave room for improvement with respect to recovering linguistic head assignment even on the Penn WSJ corpus; in fact, we find that the head-assignments according to the Magerman-Collins rules correspond only in 85% of the cases to dependencies such as annotated in PARC 700 Dependency Bank (see section 5).

Automatic methods for identifying heads are therefore of interest, both for practical and more fundamental linguistic reasons. In this paper we investigate possible ways of finding heads based on lexicalized tree structures that can be extracted from an available treebank. The starting point of our approach is the observation that a head-annotated treebank (obeying the constraint that every nonterminal node has exactly one daughter marked as head) defines a unique lexicalized tree substitution grammar (obeying the constraint that every elementary tree has exactly one lexical anchor). This allows us to go back and forth between

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 701–709,
Athens, Greece, 30 March – 3 April 2009.

the two representations, and define objective functions for the unsupervised learning of head assignments in terms of features of the implicit Lexicalized Tree Substitution Grammars.

Using this grammar formalism (LTSGs) we will investigate which objective functions we should optimize for recovering heads. Should we try to reduce uncertainty about the grammatical frames that can be associated with a particular lexical item? Or should we assume that linguistic head assignments are based on the occurrence frequencies of the productive units they imply?

We present two new algorithms for unsupervised recovering of heads – entropy minimization and a greedy technique we call "familiarity maximization" – that can be seen as ways to operationalize these last two linguistic intuitions. Both algorithms are *unsupervised*, in the sense that they are trained on data without head annotations, but both take labeled phrase-structure trees as input.

Our work fits well with several recent approaches aimed at completely unsupervised learning of the key aspects of syntactic structure: lexical categories (Schütze, 1993), phrase-structure (Klein and Manning, 2002; Seginer, 2007), phrasal categories (Borensztajn and Zuidema, 2007; Reichart and Rappoport, 2008) and dependencies (Klein and Manning, 2004).

For the specific task addressed in this paper – assigning heads in treebanks – we only know of one earlier paper: Chiang and Bikel (2002). These authors investigated a technique for identifying heads in constituency trees based on maximizing likelihood, using EM, under a Tree Insertion Grammar (TIG)model[1]. In this approach, headedness in some sense becomes a *state-split*, allowing for grammars that more closely match empirical distributions over trees. The authors report somewhat disappointing results, however: the automatically induced head-annotations do not lead to significantly more accurate parsers than simple leftmost or rightmost head assignment schemes[2].

In section 2 we define the grammar model we will use. In section 3 we describe the head-assignment algorithms. In section 4, 5 and 6 we then describe our evaluations of these algorithms.

[1]The space over the possible head assignments that these authors consider – essentially regular expressions over CFG rules – is more restricted than in the current work where we consider a larger "domain of locality".

[2]However, the authors' approach of using EM for inducing latent information in treebanks has led to extremely accurate constituency parsers, that neither make use of nor produce headedness information; see (Petrov et al., 2006)

2 Lexicalized Tree Grammars

In this section we define Lexicalised Tree Substitution Grammars (LTSGs) and show how they can be read off unambiguously from a head-annotated treebank. LTSGs are best defined as a restriction of the more general Probabilistic Tree Substitution Grammars, which we describe first.

2.1 Tree Substitution Grammars

A tree substitution grammar (TSG) is a 4-tuple $\langle V_n, V_t, S, T\rangle$ where V_n is the set of nonterminals; V_t is the set of of terminals; $S \in V_n$ is the start symbol; and T is the set of elementary trees, having root and internal nodes in V_n and leaf nodes in $V_n \cup V_t$. Two elementary trees α and β can be combined by means of the substitution operation $\alpha \circ \beta$ to produce a new tree, only if the root of β has the same label of the leftmost nonterminal leaf of α. The combined tree corresponds to α with the leftmost nonterminal leaf replaced with β. When the tree resulting from a series of substitution operations is a complete parse tree, i.e. the root is the start symbol and all leaf nodes are terminals, we define the sequence of the elementary trees used as a *complete derivation*.

A probabilistic TSG defines a probabilistic space over the set of elementary trees: for every $\tau \in T$, $P(\tau) \in [0, 1]$ and $\sum_{\tau':r(\tau')=r(\tau)} P(\tau') = 1$, where $r(\tau)$ returns the root node of τ. Assuming subsequent substitutions are stochastically independent, we define the probability of a derivation as the product of the probability of its elementary trees. If a derivation d consists of n elementary trees $\tau_1 \circ \tau_2 \circ \ldots \circ \tau_n$, we have:

$$P(d) = \prod_{i=1}^{n} P(\tau_i) \quad (1)$$

Depending on the set T of elementary trees, we might have different derivations producing the same parse tree. For any given parse tree t, we define $\delta(t)$ as the set of its derivations licensed by the grammar. Since any derivation $d \in \delta(t)$ is a possible way to construct the parse tree, we will compute the probability of a parse tree as the sum of the probabilities of its derivations:

$$P(t) = \sum_{d \in \delta(t)} \prod_{\tau \in d} P(\tau) \quad (2)$$

Lexicalized Tree Substitution Grammars are defined as TSGs with the following contraint on the set of elementary trees T: every τ in T must have at least one terminal (the *lexical anchor*) among its leaf nodes. In this paper, we are only concerned with single-anchored LTSGs, in which all elementary trees have *exactly* one lexical anchor. Like TSGs, LTSGs have a weak generative capacity that is context-free; but whereas PTSGs are both probabilistically and in terms of strong generative capacity richer than PCFGs (Bod, 1998), LTSG are more restricted (Joshi and Schabes, 1991). This limits the usefulness of LTSGs for modeling the full complexity of natural language syntax; however, computationally, LTSGs have many advantages over richer formalisms and for the current purposes represent a useful compromise between linguistic adequacy and computational complexity.

2.2 Extracting LTSGs from a head-annotated corpus

In this section we will describe a method for assigning to each word token that occurs in the corpus a unique elementary tree. This method depends on the annotation of heads in the treebank, such as for instance provided for the Penn Treebank by the Magerman-Collins head-percolation rules. We adopt the same constraint as used in this scheme, that each nonterminal node in every parse tree must have exactly one of its children annotated as head. Our method is similar to (Chiang, 2000), but is even simpler in ignoring the distinction between arguments and adjuncts (and thus the sister-adjunction operation). Figure 1 shows an example parse tree enriched with head-annotation: the suffix -H indicates that the specific node is the head of the production above it.

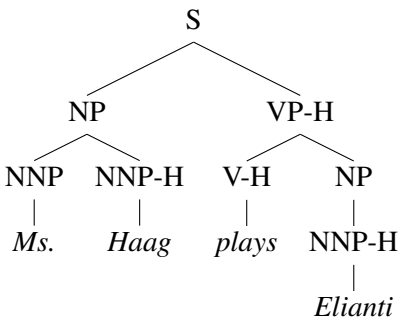


Figure 1: Parse tree of the sentence *"Ms. Haag plays Elianti"* annotated with head markers.

Once a parse tree is annotated with head markers in such a manner, we will be able to extract for every leaf its *spine*. Starting from each lexical production we need to move upwards towards the root on a path of head-marked nodes until we find the first internal node which is not marked as head or until we reach the root of the tree. In the example above, the verb of the sentence *"plays"* is connected through head-marked nodes to the root of the tree. In this way we can extract the 4 spines from the parse tree in figure 1, as shown in figure 2.

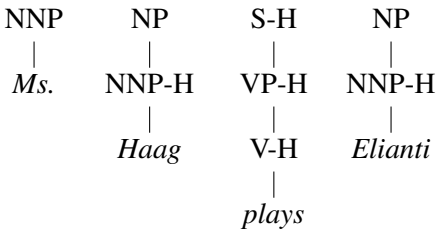


Figure 2: The lexical spines of the tree in fig. 1.

It is easy to show that this procedure yields a unique spine for each of its leaves, when applied to a parse tree where all nonterminals have a single head-daughter and all terminals are generated by a unary production. Having identified the spines, we convert them to elementary trees, by completing every internal node with the other daughter nodes not on the spine. In this way we have defined a way to obtain a derivation of any parse tree composed of lexical elementary trees. The 4 elementary trees completed from the previous paths are in figure 3 with the substitution sites marked with ⇓.

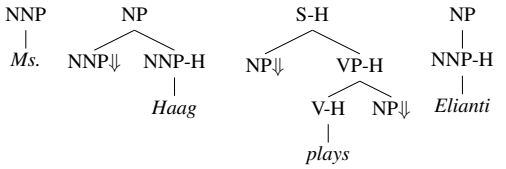


Figure 3: The extracted elementary trees.

3 Head Assignment Algorithms

We investigate two novel approaches to automatically assign head dependencies to a training corpus where the heads are not annotated: *entropy minimization* and *familiarity maximization*. The baselines for our experiments will be given by the Magerman and Collins scheme together with the random, the leftmost daughter, and the rightmost daughter-based assignments.

3.1 Baselines

The Magerman-Collins scheme, and very similar versions, are well-known and described in detail elsewhere (Magerman, 1995; Collins, 1999; Yamada and Matsumoto, 2003); here we just mention that it is based on a number of heuristic rules that only use the labels of nonterminal nodes and the ordering of daughter nodes. For instance if the root label of a parse tree is S, the head-percolation scheme will choose to assign the head marker to the first daughter from the left, labeled with TO. If no such label is present, it will look for the first IN. If no IN is found, it will look for the first VP, and so on. We used the freely available software "Treep" (Chiang and Bikel, 2002) to annotate the Penn WSJ treebank with heads.

We consider three other baselines, that are applicable to other treebanks and other languages as well: RANDOM, where, for every node in the treebank, we choose a random daughter to be marked as head; LEFT, where the leftmost daughter is marked; and RIGHT, where the rightmost daughter is marked.

3.2 Minimizing Entropy

In this section we will describe an entropy based algorithm, which aims at learning the simplest grammar fitting the data. Specifically, we take a "supertagging" perspective (Bangalore and Joshi, 1999) and aim at reducing the uncertainty about which elementary tree (supertag) to assign to a given lexical item. We achieve this by minimizing an objective function based on the general definition of entropy in information theory.

The entropy measure that we are going to describe is calculated from the bag of lexicalized elementary trees T extracted from a given training corpus of head annotated parse trees. We define T_l as a discrete stochastic variable, taking as values the elements from the set of all the elementary trees having l as lexical anchor $\{\tau_{l_1}, \tau_{l_2}, \ldots, \tau_{l_n}\}$. T_l thus takes n possible values with specific probabilities; its entropy is then defined as:

$$H(T_l) = -\sum_{i=1}^{n} p(\tau_{l_i}) \log_2 p(\tau_{l_i}) \qquad (3)$$

The most intuitive way to assign probabilities to each elementary tree is considering its relative frequency in T. If $f(\tau)$ is the frequency of the fragment τ and $f(l)$ is the total frequency of fragments with l as anchor we will have:

$$p(\tau_{l_j}) = \frac{f(\tau_{l_j})}{f(lex(\tau_{l_j}))} = \frac{f(\tau_{l_j})}{\sum_{i=1}^{n} f(\tau_{l_i}))} \qquad (4)$$

We will then calculate the entropy $H(T)$ of our bag of elementary trees by summing the entropy of each single discrete stochastic variable T_l for each choice of l:

$$H(T) = \sum_{l=1}^{|\mathscr{L}|} H(T_l) \qquad (5)$$

In order to minimize the entropy, we apply a *hill-climbing* strategy. The algorithm starts from an already annotated tree-bank (for instance using the RANDOM annotator) and iteratively tries out a random change in the annotation of each parse tree. Only if the change reduces the entropy of the entire grammar it is kept. These steps are repeated until no further modification which could reduce the entropy is possible. Since the entropy measure is defined as the sum of the function $p(\tau) \log_2 p(\tau)$ of each fragment τ, we do not need to re-calculate the entropy of the entire grammar, when modifying the annotation of a single parse tree. In fact:

$$\begin{aligned} H(T) &= -\sum_{l=1}^{|\mathscr{L}|} \sum_{i=1}^{n} p(\tau_{l_i}) \log_2 p(\tau_{l_i}) \\ &= -\sum_{j=1}^{|T|} p(\tau_j) \log_2 p(\tau_j) \end{aligned} \qquad (6)$$

For each input parse tree under consideration, the algorithm selects a non-terminal node and tries to change the head annotation from its current head-daughter to a different one. As an example, considering the parse tree of figure 1 and the internal node NP (the leftmost one), we try to annotate its leftmost daughter as the new head. When considering the changes that this modification brings on the set of the elementary trees T, we understand that there are only 4 elementary trees affected, as shown in figure 4.

After making the change in the head annotation, we just need to decrease the frequencies of the *old trees* by one unit, and increase the ones of the *new trees* by one unit. The change in the entropy of our grammar can therefore be computed by calculating the change in the partial entropy of these four

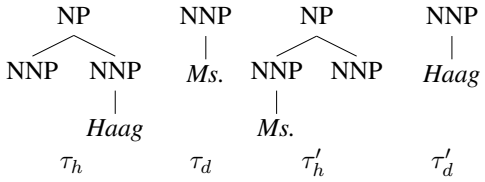


Figure 4: Lexical trees considered in the ENTROPY algorithm when changing the head assingnment from the second NNP to the first NNP of the leftmost NP node of figure 1. τ_h is the *old head tree*; τ_d the *old dependent tree*; τ'_d the *new dependent tree*; τ'_h the *new head tree*.

elementary trees before and after the change. If such change results in a lower entropy of the grammar, the new annotation is kept, otherwise we go back to the previous annotation. Although there is no guarantee our algorithm finds the global minimum, it is very efficient and succeeds in drastically minimize the entropy from a random annotated corpus.

3.3 Maximizing Familiarity

The main intuition behind our second method is that we like to assign heads to a tree t in such a way that the elementary trees that we can extract from t are frequently observed in other trees as well. That is, we like to use elementary trees which are general enough to occur in many possible constructions.

We start with building the bag of all one-anchor lexicalized elementary trees from the training corpus, consistent with *any* annotation of the heads. This operation is reminiscent of the extraction of all subtrees in Data-Oriented Parsing (Bod, 1998). Fortunately, and unlike DOP, the number of possible lexicalised elementary trees is not exponential in sentence length n, but polynomial: it is always smaller than n^2 if the tree is binary branching.

Next, for each node in the treebank, we need to select a specific lexical anchor, among the ones it dominates, and annotate the nodes in the spine with head annotations. Our algorithm selects the lexical anchor which maximizes the frequency of the implied elementary tree in the bag of elementary trees. In figure 5, algorithm 1 (right) gives the pseudo-code for the algorithm, and the tree (left) shows an example of its usage.

3.4 Spine and POS-tag reductions

The two algorithms described in the previous two sections are also evaluated when performing two possible generalization operations on the elementary trees, which can be applied both alone or in combination:

- in the *spine reduction*, lexicalized trees are transformed to their respective spines. This allows to merge elementary trees that are slightly differing in argument structures.
- in the *POStag reduction*, every lexical item of every elementary tree is replaced by its POStag category. This allows for merging elementary trees with the same internal structure but differing in lexical production.

4 Implementation details

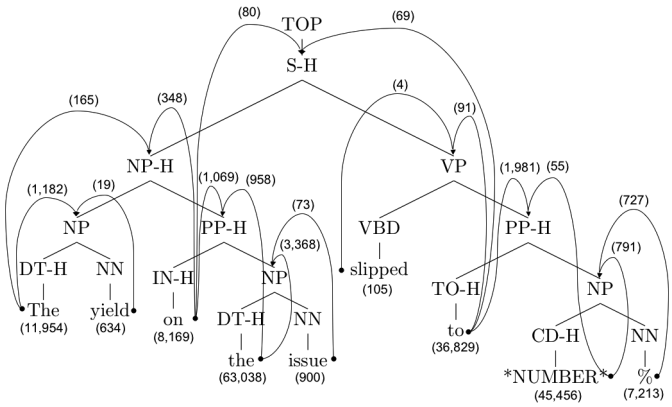
4.1 Using CFGs for TSG parsing

When evaluating parsing accuracy of a given LTSG, we use a CKY PCFG parser. We will briefly describe how to set up an LTSG parser using the CFG formalism. Every elementary tree in the LTSG should be treated by our parser as a unique block which cannot be further decomposed. But to feed it to a CFG-parser, we need to break it down into trees of depth 1. In order to keep the integrity of every elementary tree we will assign to its internal node a unique label. We will achieve this by adding "@i" to each i-th internal node encountered in T.

Finally, we read off a PCFG from the elementary trees, assigning to each PCFG rule a weight proportional to the weight of the elementary tree it is extracted from. In this way the PCFG is equivalent to the original LTSG: it will produce exactly the same derivation trees with the same probabilities, although we would have to sum over (exponentially) many derivations to obtain the correct probabilities of a parse tree (*derived tree*). We approximate parse probability by computing the n-best derivations and summing over the ones that yield the same parse tree (by removing the "@i"-labels). We then take the parse tree with highest probability as best parse of the input sentence.

4.2 Unknown words and smoothing

We use a simple strategy to deal with unknown words occurring in the test set. We replace all the words in the training corpus occurring once, and all the unknown words in the test set, with a special *UNKNOWN* tag. Moreover we replace all the numbers in the training and test set with a special *NUMBER* tag.



Algorithm 1: $MaximizeFamiliarity(N)$

Input: a non-terminal node N of a parsetree.
begin
  $L = null$; $MAX = -1$;
  foreach *leaf* l *under* N **do**
    τ_l^N = lex. tree rooted in N and anchored in l;
    F = frequency of τ_l^N;
    if $F > MAX$ **then**
      $L = l$; $MAX = F$;
  Mark all nodes in the path from N to L with heads;
  foreach *substitution site* N_i *of* τ_L^N **do**
    $MaximizeFamiliarity(N_i)$;
end

Figure 5: Left: example of a parse tree in an instantiation of the "Familiarity" algorithm. Each arrow, connecting a word to an internal node, represents the elementary tree anchored in that word and rooted in that internal node. Numbers in parentheses give the frequencies of these trees in the bag of subtrees collected from WSJ20. The number below each leaf gives the total frequency of the elementary trees anchored in that lexical item. Right: pseudo-code of the "Familiarity" algorithm.

Even with unknown words treated in this way, the lexicalized elementary trees that are extracted from the training data are often too specific to parse all sentences in the test set. A simple strategy to ensure full coverage is to smooth with the treebank PCFG. Specifically, we add to our grammars all CFG rules that can be extracted from the training corpus and give them a small weight proportional to their frequency[3]. This in general will ensure coverage, i.e. that all the sentences in the test set can be successfully parsed, but still prioritizing lexicalized trees over CFG rules[4].

4.3 Corpora

The evaluations of the different models were carried out on the Penn Wall Street Journal corpus (Marcus et al., 1993) for English, and the Tiger treebank (Brants et al., 2002) for German. As gold standard head annotations corpora, we used the Parc 700 Dependency Bank (King et al., 2003) and the Tiger Dependency Bank (Forst et al., 2004), which contain independent reannotations of extracts of the WSJ and Tiger treebanks.

5 Results

We evaluate the head annotations our algorithms find in two ways. First, we compare the head annotations to gold standard manual annotations of heads. Second, we evaluate constituency parsing performance using an LTSG parser (trained on the various LTSGs), and a state-of-the-art parser (Bikel, 2004).

5.1 Gold standard head annotations

Table 1 reports the performance of different algorithms against gold standard head annotations of the WSJ and the Tiger treebank. These annotations were obtained by converting the dependency structures of the PARC corpus (700 sentences from section 23) and the Tiger Dependency Bank (2000 sentences), into head annotations[5]. Since the algorithm doesn't guarantee that the recovered head annotations always follow the one-head-per-node constraint, when evaluating the accuracy of head annotations of different algorithms, we exclude the cases in which in the gold corpus no head or multiple heads are assigned to the daughters of an internal node[6], as well as cases in which an internal node has a single daughter.

In the evaluation against gold standard dependencies for the PARC and Tiger dependency banks, we find that the FAMILIARITY algorithm when run with POStags and Spine conversion obtains around 74% recall for English and 69% for German. The different scores of the RANDOM assignment for the two languages can be explained

[3] In our implementation, each CFG rule frequency is divided by a factor 100.

[4] In this paper, we prefer these simple heuristics over more elaborate techniques, as our goal is to compare the merits of the different head-assignment algorithms.

[5] This procedure is not reported here for reasons of space, but it is available for other researchers (together with the extracted head assignments) at `http://staff.science.uva.nl/~fsangati`.

[6] After the conversion, the percentage of incorrect heads in PARC 700 is around 9%; in Tiger DB it is around 43%.

by their different branching factors: trees in the German treebank are typically more flat than those in the English WSJ corpus. However, note that other settings of our two annotation algorithms do not always obtain better results than random.

When focusing on the Tiger results, we observe that the RIGHT head assignment recall is much better than the LEFT one. This result is in line with a classification of German as a predominantly head-final language (in contrast to English). More surprisingly, we find a relatively low recall of the head annotation in the Tiger treebank, when compared to a gold standard of dependencies for the same sentences as given by the Tiger dependency bank. Detailed analysis of the differences in head assigments between the two approaches is left for future work; for now, we note that our best performing algorithm approaches the inter-annotation-scheme agreement within only 10 percentage points[7].

5.2 Constituency Parsing results

Table 2 reports the parsing performances of our LTSG parser on different LTSGs extracted from the WSJ treebank, using our two heuristics together with the 4 baseline strategies (plus the result of a standard treebank PCFG). The parsing results are computed on WSJ20 (WSJ sentences up to length 20), using sections 02-21 for training and section 22 for testing.

We find that all but one of the head-assignment algorithms lead to LTSGs that without any fine-tuning perform better than the treebank PCFG. On this metric, our best performing algorithm scores 4 percentage points higher than the Magerman-Collins annotation scheme (a 19% error reduction). The poor results with the RIGHT assignment, in contrast with the good results with the LEFT baseline (performing even better than the Magerman-Collins assignments), are in line with the linguistic tradition of listing English as a predominantly head-initial language. A surprising result is that the RANDOM-assignment gives the best performing LTSG among the baselines. Note, however, that this strategy leads to much wieldier grammars; with many more elementary trees than for instance the left-head assignment, the RANDOM strategy is apparently better equipped to parse novel sentences. Both the FAMILIARITY and the ENTROPY strategy are at the level of the random-head assignment, but do in fact lead to much more compact grammars.

We have also used the same head-enriched treebank as input to a state-of-the-art constituency parser[8] (Bikel, 2004), using the same training and test set. Results, shown in table 3, confirm that the differences in parsing success due to different head-assignments are relatively minor, and that even RANDOM performs well. Surprisingly, our best FAMILIARITY algorithm performs as well as the Collins-Magerman scheme.

	LFS	**UFS**	**|T|**
PCFG	78.23	82.12	-
RANDOM	**82.70**	**85.54**	**64k**
LEFT	80.05	83.21	46k
Magerman-Collins	79.01	82.67	54k
RIGHT	73.04	77.90	49k
FAMILIARITY	**84.44**	**87.22**	**42k**
ENTROPY-POStags	82.81	85.80	64k
FAMILIARITY-Spine	82.67	85.35	47k
ENTROPY-POStags-Spine	82.64	85.55	64k

Table 2: Parsing accuracy on WSJ20 of the LTSGs extracted from various head assignments, when computing the most probable derivations for every sentence in the test set. The Labeled F-Score (LFS) and unlabeled F-Score (UFS) results are reported. The final column gives the total number of extracted elementary trees (in thousands).

	LFS	**UFS**
Magerman-Collins	**86.20**	**88.35**
RANDOM	84.58	86.97
RIGHT	81.62	84.41
LEFT	81.13	83.95
FAMILIARITY-POStags	**86.27**	**88.32**
FAMILIARITY-POStags-Spine	85.45	87.71
FAMILIARITY-Spine	84.41	86.83
FAMILIARITY	84.28	86.53

Table 3: Evaluation on WSJ20 of various head assignments on Bikel's parser.

[7] We have also used the various head-assignments to convert the treebank trees to dependency structures, and used these in turn to train a dependency parser (Nivre et al., 2005). Results from these experiments confirm the ordering of the various *unsupervised* head-assignment algorithms. Our best results, with the FAMILIARITY algorithm, give us an Unlabeled Attachment Score (UAS) of slightly over 50% against a gold standard obtained by applying the Collins-Magerman rules to the test set. This is much higher than the three baselines, but still considerably worse than results based on *supervised* head-assignments.

[8] Although we had to change a small part of the code, since the parser was not able to extract heads from an enriched treebank, but it was only compatible with rule-based assignments. For this reason, results are reported only as a base of comparison.

Gold = PARC 700	% correct	Gold = Tiger DB	% correct
Magerman-Collins	**84.51**	**Tiger TB Head Assignment**†	**77.39**
LEFT	47.63	RIGHT	52.59
RANDOM	43.96	RANDOM	38.66
RIGHT	40.70	LEFT	18.64
FAMILIARITY-POStags-Spine	**74.05**	**FAMILIARITY-POStags-Spine**	**68.88**
FAMILIARITY-POStags	51.10	FAMILIARITY-POStags	41.74
ENTROPY-POStags-Spine	43.23	ENTROPY-POStags-Spine	37.99
FAMILIARITY-Spine	39.68	FAMILIARITY	26.08
FAMILIARITY	37.40	FAMILIARITY-Spine	22.21

Table 1: Percentage of correct head assignments against gold standard in Penn WSJ and Tiger.
† The Tiger treebank already comes with built-in head labels, but not for all categories. In this case the score is computed only for the internal nodes that conform to the one head per node constraint.

6 Conclusions

In this paper we have described an empirical investigation into possible ways of enriching corpora with head information, based on different linguistic intuitions about the role of heads in natural language syntax. We have described two novel algorithms, based on entropy minimization and familiarity maximization, and several variants of these algorithms including POS-tag and spine reduction.

Evaluation of head assignments is difficult, as no widely agreed upon gold standard annotations exist. This is illustrated by the disparities between the (widely used) Magerman-Collins scheme and the Tiger-corpus head annotations on the one hand, and the "gold standard" dependencies according to the corresponding Dependency Banks on the other. We have therefore not only evaluated our algorithms against such gold standards, but also tested the parsing accuracies of the implicit lexicalized grammars (using three different parsers). Although the ordering of the algorithms on performance on these various evaluations is different, we find that the best performing strategies in all cases and for two different languages are with variants of the "familiarity" algorithm.

Interestingly, we find that the parsing results are consistently better for the algorithms that keep the full lexicalized elementary trees, whereas the best matches with gold standard annotations are obtained by versions that apply the POStag and spine reductions. Given the uncertainty about the gold standards, the possibility remains that this reflects biases towards the most general headedness-rules in the annotation practice rather than a linguistically real phenomenon.

Unsupervised head assignment algorithms can be used for the many applications in NLP where information on headedness is needed to convert constituency trees into dependency trees, or to extract head-lexicalised grammars from a constituency treebank. Of course, it remains to be seen which algorithm performs best in any of these specific applications. Nevertheless, we conclude that among currently available approaches, i.e., our two algorithms and the EM-based approach of (Chiang and Bikel, 2002), "familiarity maximization" is the most promising approach for automatic assignments of heads in treebanks.

From a linguistic point of view, our work can be seen as investigating ways in which distributional information can be used to determine headedness in phrase-structure trees. We have shown that lexicalized tree grammars provide a promising methodology for linking alternative head assignments to alternative dependency structures (needed for deeper grammatical structure, including e.g., argument structure), as well as to alternative derivations of the same sentences (i.e. the set of lexicalized elementary trees need to derive the given parse tree). In future work, we aim to extend these results by moving to more expressive grammatical formalisms (e.g., tree adjoining grammar) and by distinguishing adjuncts from arguments.

Acknowledgments We gratefully acknowledge funding by the Netherlands Organization for Scientific Research (NWO): FS is funded through a Vici-grant "Integrating Cognition" (277.70.006) to Rens Bod and WZ through a Veni-grant "Discovering Grammar" (639.021.612). We thank Rens Bod, Yoav Seginer, Reut Tsarfaty and three anonymous reviewers for helpful comments, Thomas By for providing us with his dependency bank and Joakim Nivre and Dan Bikel for help in adapting their parsers to work with our data.

References

S. Bangalore and A.K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

D.M. Bikel. 2004. Intricacies of Collins' Parsing Model. *Computational Linguistics*, 30(4):479–511.

R. Bod. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI, Stanford, CA.

G. Borensztajn, and W. Zuidema. 2007. Bayesian Model Merging for Unsupervised Constituent Labeling and Grammar Induction. Technical Report, ILLC.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

T. By. 2007. Some notes on the PARC 700 dependency bank. *Natural Language Engineering*, 13(3):261–282.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the fourteenth national conference on artificial intelligence*, Menlo Park. AAAI Press/MIT Press.

D. Chiang and D.M. Bikel. 2002. Recovering latent information in treebanks. *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7.

D. Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the ACL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

G. Corbett, N. Fraser, and S. McGlashan, editors. 2006. *Heads in Grammatical Theory*. Cambridge University Press.

M. Forst, N. Bertomeu, B. Crysmann, F. Fouvry, S. Hansen-Schirra, and V. Kordoni. 2004. Towards a dependency-based gold standard for German parsers.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Comput. Linguist.*, 33(3):355–396.

A.K. Joshi and Y. Schabes. 1991. Tree-adjoining grammars and lexicalized grammars. Technical report, Department of Computer & Information Science, University of Pennsylvania.

T. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. 2003. The PARC 700 dependency bank.

D. Klein and C.D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the ACL*.

D. Klein and C.D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the ACL*.

D.M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the ACL*.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

J. Nivre and J. Hall. 2005. MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. In Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT2005), pages 137–148.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson,S. Riedel, and D. Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task.*, June.

J. Nivre. 2007. Inductive Dependency Parsing. *Computational Linguistics*, 33(2).

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings ACL-COLING'06*, pages 443–440.

R. Reichart and A. Rappoport. 2008. Unsupervised Induction of Labeled Parse Trees by Clustering with Syntactic Features. In *Proceedings Coling*.

H. Schütze. 1993. Part-of-speech induction from scratch. In *Proceedings of the 31st annual meeting of the ACL*.

Y. Seginer 2007. *Learning Syntactic Structure*. Ph.D. thesis, University of Amsterdam.

H. Yamada, and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the Eighth International Workshop on Parsing Technologies.* Nancy, France.