Discriminant Ranking for Efficient Treebanking

Yi Zhang **Valia Kordoni**
LT-Lab, German Research Center for Artificial Intelligence (DFKI GmbH)
Department of Computational Linguistics, Saarland University
{yzhang,kordoni}@coli.uni-sb.de

Abstract

Treebank annotation is a labor-intensive and time-consuming task. In this paper, we show that a simple statistical ranking model can significantly improve treebanking efficiency by prompting human annotators, well-trained in disambiguation tasks for treebanking but not necessarily grammar experts, to the most relevant linguistic disambiguation decisions. Experiments were carried out to evaluate the impact of such techniques on annotation efficiency and quality. The detailed analysis of outputs from the ranking model shows strong correlation to the human annotator behavior. When integrated into the treebanking environment, the model brings a significant annotation speed-up with improved inter-annotator agreement.[†]

1 Introduction

The development of a large-scale treebank (Marcus et al., 1993; Hajič et al., 2000; Brants et al., 2002) with rich syntactic annotations is a highly rewarding task. But the huge amount of manual labor required for the annotation task itself, as well as the difficulties in standardizing linguistic analyses, results in long development cycles of such valuable language resources, which typically amounts to years or even decades. Despite the profound scientific and practical value of detailed syntactic treebanks, the requirement and necessity for long-term commitment raises the risk cost of such projects, a fact which often makes them not feasible in many current economical environments.

In recent years, computational grammars have been employed to assist the construction of such language resources. A typical development model involves a parser which generates candidate analyses, and human annotators who manually identify the desired tree structure. This treebanking method dramatically reduces the cost of training annotators, for they are not required to spontaneously produce linguistic solutions to various phenomena. Instead, they are trained to associate their language intuition with specific linguistically-relevant decisions. How to select and carefully present such decisions to the annotators is thus crucial for achieving high annotation speed and quality. On the other hand, for large treebanking projects, parallel annotation with multiple annotators is usually necessary. Inter-annotator agreement is a crucial quality measure in such cases. But improvements on annotation speed should not be achieved at expense of the quality of the treebank.

With both speed and quality in mind, a good treebank annotation method should acknowledge the complexity of the decision-making process; for instance, the same tree can be disambiguated by different sets of individual decisions which are mutually dependent. The annotation method should also strive to create a distraction-free environment for annotators who can then focus on making the judgments. To this effect, we present a simple statistical model that learns from the annotation history, and offers a ranking of disambiguation decisions from the most to the least relevant

[†]The first author thanks the German Excellence Cluster of Multimodal Computing and Interaction for the support of the work.

ones, which enables well-trained human annotators to speed-up treebanking without compromising on the quality of the linguistic decisions guiding the annotation task.

The remaining of this paper is structured as follows: Section 2 gives an overview of the difficulties in syntactic annotation, and the potential ways of improving the annotation efficiency without damaging the quality; Section 3 presents the new annotation method which is based on a statistical discriminant ranking model; Sections 4 and 5 describe the setup and results of a series of annotation experiments; Section 6 concludes the paper and proposes future research directions.

2 Background

Large-scale full syntactic annotation has for quite some time been approached with mixed feelings by researchers. On the one hand, detailed syntactic annotation serves as a basis for corpus-linguistic study and data-driven NLP methods. Especially, when combined with popular supervised machine learning methods, richly annotated language resources, like, for instance, treebanks, play a key role in modern computational linguistics. The public availability of large-scale treebanks in recent years has stimulated the blossoming of data-driven approaches to syntactic and semantic parsing.

On the other hand, though, the creation of detailed syntactic structures turns out to be an extremely challenging task. From the choice of the appropriate linguistic framework and the design of the annotation scheme to the choice of the text source and the working protocols on the syncronization of the parallel development, as well as the quality assurance, none of these steps in the entire annotation procedure is considered a solved issue. Given the vast design choices, very few of the treebanking projects have made it through all these difficult annotation stages. Even the most outstanding projects have not been completed without receiving criticisms.

Our treebanking project is no exception. The aim of the project is to provide annotations of the Wall Street Journal (henceforward `WSJ`) sections of the Penn Treebank (henceforward `PTB` (Marcus et al., 1993)) with the help of the English Resource Grammar (henceforward `ERG`; (Flickinger, 2002)), a hand-written large-scale and wide-coverage grammar of English in the framework of Head-Driven Phrase Structure Grammar (`HPSG`; (Pollard and Sag, 1994)). Such annotations are very rich linguistically, since apart from syntax they also incorporate semantic information. The annotation cycle is organized into iterations of parsing, treebanking in the sense of disambiguating syntactic and semantic analyses of the various linguistic phenomena contained in the corpus, error analysis and grammar/treebank update cycles. That is, sentences from the `WSJ` are first parsed with the `PET` parser (Callmeier, 2001), an efficient unification-based parser, using the `ERG`. The parsing results are then manually disambiguated by human annotators. However, instead of considering individual trees, the annotation process is mostly invested on binary decisions which are made on either accepting or rejecting constructions or lexical types. Each of such decisions, called discriminants, as we will also see in the following, reduces the number of the trees satisfying the constraints. The process is presented in the next section in more detail. What should, though, be clear is that the aforementioned multi-cycle annotation procedure is as time-consuming and human-error prone as any other, despite the fact that at the center of the entire annotation cycle lies a valuable linguistic resource, which has been developed with a lot of effort over many years, namely the `ERG`. For the first period of this project, we have established an average speed of 40 sentences per annotator hour, meaning a total of $\sim$1200 hours of annotation for the entire `WSJ`. Including the long training period at the beginning of the project, and periodical grammar and treebank updates, the project period is roughly two years with two part-time annotators employed.

3 Statistical Discriminant Ranking

3.1 Discriminants & Decisions

One common characteristic of modern treebanking efforts – especially, in so-called dynamic treebanking platforms (cf., for instance, (Oepen et al., 2002) and `http://redwoods.stanford.edu`), like the one we are describing and referring to extensively in the following, is that the candidate trees are constructed automatically by the grammar, and then manually disambiguated by human annotators. In doing so, linguistically rich annotation is built efficiently with minimum manual labor. In order to further improve the manual disambiguation efficiency, systems like `[incr tsdb()]` (Oepen, 2001) compute the difference between candidate analyses. Instead of looking at the huge parse forest, the treebank annotators select or reject the features that distinguish between different parses, until no ambiguity remains (either one analysis is accepted from the parse forest, or all of them are rejected). The number of decisions for each sentence is normally around $log_2 n$ where n is the total number of candidate trees. For a sentence with 5000 candidate readings, only about 12 treebanking decisions are required for a complete disambiguation. A similar method was also proposed in (Carter, 1997).

Formally, an attribute that distinguishes between different parses is called a *discriminant*. For Redwoods-style treebanks, this is extracted either from the syntactic derivation trees or the semantic representations (in the form of Minimal Recursion Semantics (MRS; (Copestake et al., 2005))).

Figure 1 shows an example graphical annotation interface. At the top of the window, a list of action buttons shows the operations permitted on the sentence level. Then the sentence in its original PTB bracket format is shown. 15 : 0 indicates that at the current disambiguation state, 15 trees remain to be disambiguated while 0 has been eliminated. On the left large panel, the candidate trees are shown in their simplified phrase-structure representation. Note that the actual HPSG analyses are not shown in the screenshot and can be displayed on request. On the right large panel, the list of effective discriminants (see Section 3.2) up to this disambiguation state is shown. The highlighted discriminant in Figure 1 suggests a possibility of constructing the entire sentence by choosing a subject-head rule (SUBJH), taking *"ms. Haag"* as the subject and *"plays Elianti."* as the head daughter. When the discriminant is clicked, the annotator can say *yes* or *no* to it, hence narrowing the remaining trees to the *In Parses* or *Out Parses*. The *unknown* button is used to mark the uncertainties and is rarely used.

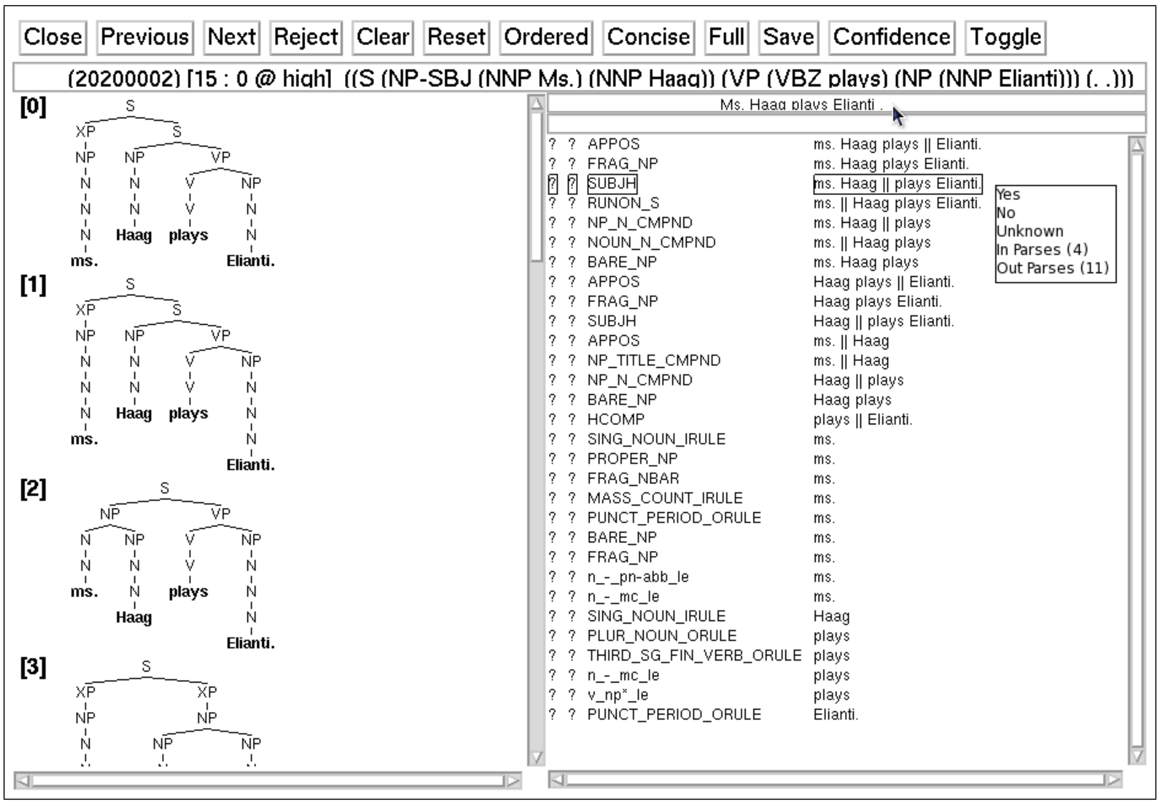
Note that in this interface, the discriminants are sorted in descending order according to their length, meaning that the discriminants related to higher level constructions are shown before the lexical type choices. When up to 500 parses are stored in the forest, the average number of discriminants per forest is about 100. Scanning through the long list manually can be time-consuming and distracting.

Kordoni and Zhang (2009) show that annotators tend to start with the decisions with the most certainty, and delay the "hard" decisions as much as possible. As the decision progresses, many of the "hard" discriminants will receive an inferred value from the certain decisions. Our annotation guideline only describes specific decisions. The order in which discriminants are chosen is left underspecified and very much depends on personal styles. In practice, we see that our annotators gradually developed complex strategies which involve both top-down and bottom-up pruning.

One potential drawback of such a discriminant-based treebanking method is that the process is very sensitive to decision errors. One wrong judgment can rule out the correct tree and ruin the analysis of the sentence. In such a case, the annotators usually resort to backtracking to previous decisions they had made. To compensate for this, we ask our annotators to double-check the treebanked analysis before saving the disambiguation result. And in case of doubt, they are instructed to avoid ambivalent decisions as much as possible.

3.2 Maximum-Entropy-Based Discriminant Ranking Model

Suppose for a sentence ω, a parse forest Y was generated by the grammar. Note that for efficiency reasons, the parse forest might have been trimmed to only contain up to n top readings ranked by the parse disambiguation model. For convenience, we note the parse forest Y as a set of parses $\{y_1, y_2, \dots, y_n\}$. Each discriminant d defines a binary valued function δ on the parse forest ($\delta : Y \mapsto \{0, 1\}$) , which can be interpreted as whether a parse y_i has attribute d or not. By the nature of this definition, each discriminant



```
(20200002) [15 : 0 @ high]  ((S (NP-SBJ (NNP Ms.) (NNP Haag)) (VP (VBZ plays) (NP (NNP Elianti))) (. .)))
Ms. Haag plays Elianti .

? ? APPOS                     ms. Haag plays || Elianti.
? ? FRAG_NP                   ms. Haag plays Elianti.
? ? SUBJH                     ms. Haag || plays Elianti.
? ? RUNON_S                   ms. || Haag plays Elianti.
? ? NP_N_CMPND                ms. Haag || plays
? ? NOUN_N_CMPND              ms. || Haag plays
? ? BARE_NP                   ms. Haag plays
? ? APPOS                     Haag plays || Elianti.
? ? FRAG_NP                   Haag plays Elianti.
? ? SUBJH                     Haag || plays Elianti.
? ? APPOS                     ms. || Haag
? ? NP_TITLE_CMPND            ms. || Haag
? ? NP_N_CMPND                Haag || plays
? ? BARE_NP                   Haag plays
? ? HCOMP                     plays || Elianti.
? ? SING_NOUN_IRULE           ms.
? ? PROPER_NP                 ms.
? ? FRAG_NBAR                 ms.
? ? MASS_COUNT_IRULE          ms.
? ? PUNCT_PERIOD_ORULE        ms.
? ? BARE_NP                   ms.
? ? FRAG_NP                   ms.
? ? n_-_pn-abb_le             ms.
? ? n_-_mc_le                 ms.
? ? SING_NOUN_IRULE           Haag
? ? PLUR_NOUN_ORULE           plays
? ? THIRD_SG_FIN_VERB_ORULE   plays
? ? n_-_mc_le                 plays
? ? v_np*_le                  plays
? ? PUNCT_PERIOD_ORULE        Elianti.
```

Figure 1: Screenshot of the discriminant-based treebanking graphical annotator interface

function defines a bi-partition of the parse forest. When both subsets of the partition are non-empty, i.e., there exists at least one y_p and y_q such that $\delta(y_p) = 0$ and $\delta(y_q) = 1$, the discriminant is considered *effective* on the forest Y. In the following discussion, we are only considering the set of effective discriminants D for parse forest Y.

Instead of directly predicting the outcome of disambiguation decision on each discriminant (i.e., whether the GOLD tree has discriminant function value 0 or 1), our model tries to measure the probability of a discriminant being chosen by human annotators, regardless of the *yes/no* decision. For each discriminant d, and the parse forest Y, a set of feature functions $f_1, f_2, \ldots, f_k$ receive real values, and contribute to the following log-linear model:

$$P(d|Y,D) = \frac{exp(\sum_{i=1}^{k} \lambda_i f_i(d,Y))}{\sum_{d' \in D} exp(\sum_{i=1}^{k} \lambda_i f_i(d',Y))} \tag{1}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_k$ are the parameters of the model.

To estimate these model parameters, we gather the annotation logs from our treebank annotators on the completed datasets with detailed information about each discriminant. Apart from the necessary information to reconstruct the discriminants from the forest, the log also contains the status information of i) whether the discriminant takes value 0 or 1 on the gold tree; ii) whether the human annotator has said *yes* or *no* to the discriminant. Note that the human annotator does not need to manually decide on the value of each discriminant. Whenever a new decision is made, the forest will be pruned to the subset of trees compatible with the decision. And all remaining discriminants are checked for effectiveness on the pruned forest. Discriminants which become ineffective from previous decisions are said to have received *inferred* values.

The parameters of the model are estimated by the open-source maximum entropy parameter estimation toolkit TADM[1]. For training, we use all the manually disambiguated discriminants as positive instances, and automatically inferred discriminants as negative instances.

The discriminant ranking model is applied during the manual annotation sessions. When a parse forest is loaded and the discriminants are constructed, each discriminant is assigned an (unnormalized) score $\sum_{i=1}^{k} \lambda_i f_i(d, Y)$, and the list of discriminants is sorted by descending order of the score accordingly. The scoring and sorting adds negligible additional computation on the treebanking software, and is not noticeable to the human annotators. By putting those discriminants that are potentially to be manually judged near the top of the list, the model saves manual labor on scanning through the lengthy list by filtering out ambivalent discriminants.

Note that this discriminant ranking model predicts the possibility of a discriminant being manually disambiguated. It is not modeling the specific decision that the human annotator makes on the discriminant. Including the decision outcome in the model can potentially damage the annotation quality if annotators develop a habit of overtrusting the model prediction, making the whole manual annotation pointless. A discriminant ranking model, however, only suggestively re-orders the discriminants on the presentation level, which are much safer when the annotation quality is concerned.

3.3 Feature Model for Syntactic Discriminants

In practice, there are different ways of finding discriminants from the parse forest. For instance, the `[incr tsdb()]` system supports both syntax-based and semantics-based discriminants. The syntax-based discriminants are extracted from the derivation trees of the HPSG analyses. All HPSG rule applications (unary or binary) and choices of lexical entries are picked as candidate discriminants and checked for effectiveness. The semantics-based discriminants, on the other hand, represent the differences on the semantic structures (MRS in the cases of DELPH-IN[2] grammars). With a few exceptions, many DELPH-IN HPSG treebanks choose to use the syntactic discriminants which allow human annotators to pick the low-level constructions. The above proposed ranking model works for different types of discriminants (and potentially a mixture of different discriminant types). But for the evaluation of this paper, we show the feature model designed for the syntactic discriminants only.

The syntactic discriminants record the differences between derivation trees by memorizing direct rule applications and lexical choices. Beside the rule or lexical entry name, the discriminant also records the information concerning the corresponding constituent, e.g., the category and spanning of the constituent, the parent and daughters of the constituent, etc. Furthermore, given the discriminant d and the parse forest Y, we can calculate the distribution of parses over the value of the discriminant function δ, which can be characterized by $\sum_{y \in Y} \delta(y)/|Y|$. This numeric feature indicates how many parses can be ruled out with the given discriminant.

As example, for the highlighted discriminant in Figure 1, the extracted features are listed in Table 1.

4 Experiment Setup

To test the effectiveness of the discriminant ranking models, we carried out a series of experiments, investigating their effects on both annotation speed and quality. The experiment was done in the context of our ongoing annotation project of the WSJ sections of the PTB described in Section 2. Despite sharing the source of texts, the new project aims to create an independently annotated corpus. Therefore, the trees from the PTB were not used to guide the disambiguation process. In this annotation project, two annotators (both graduate students, referred to as A and B below) are employed to manually disambiguate the parsing outputs of the ERG. For quality control and adjudication in case of disagreement, a third linguist/grammarian annotates parts of the treebank in parallel.

With the help of our annotation log files, which record in details the manual decision-making process, we trained three discriminant ranking mod-

[1] http://tadm.sourceforge.net/
[2] http://www.delph-in.net/

Feature	Possible Values	Example
discriminant type	RULE/LEX	RULE
edge position	FULL/FRONT/BACK	FULL
edge span	$length$(constituent)$/length$(sentence)	4/4
edge category	rule or lexical type name	SUBJH
level of discrimination	$\sum_{y \in Y} \delta(y)/\|Y\|$	4/15
branch splitting	$length$(left-dtr)$/length$(constituent)	2/4

Table 1: Features for syntactic discriminant ranking model and example values for the highlighted discriminant in Figure 1

els with the datasets completed so far: MODEL-A and MODEL-B trained with annotation logs from two annotators separately, and MODEL-BOTH trained jointly with data from both annotators. For each annotator's model (MODEL-A and MODEL-B), we used about 6,000 disambiguated parse forests for training. For each of these 6,000 forests, the log file contains about 600,000 effective discriminants, among which only ~6% received a manual decision.

To evaluate the treebanking speed, we have the annotators work under a distraction-free environment and record their annotation speed. The speed is averaged over several 1-hour annotation sessions. Different discriminant ranking models were used without the annotators being informed of the details of the setting.

As testing dataset, we use the texts from the PARC 700 Dependency Bank (King et al., 2003), which include 700 carefully selected sentences from the WSJ sections of the PTB. These sentences were originally chosen for the purpose of parser evaluation. Many linguistically challenging phenomena are included in these sentences, although the sentence length is shorter in average than the sentence length in the entire WSJ. The language is also less related to the financial domain specific language observed in the WSJ. We parsed the dataset with the Feb. 2009 version of the ERG, and recorded up to 500 trees per sentence (ranked by a MaxEnt parse selection model trained on previously treebanked WSJ sections).

5 Results

Although we employed a typical statistical ranking model in our system, it is difficult to directly evaluate the absolute performance of the predicted ranking. Annotators only annotate a very small subset of the discriminants, and their order is not fully specified. To compare the behavior of models trained with data annotated by different annotators, we plot the relative ranking (normalized to $[0, 1]$ for each sentence, with 0 being the highest rank and 1 the lowest) of discriminants for 50 sentences in Figure 2.

The plot shows a strong positive linear correlation between the two ranking models. The particularly strong correlation at the low and high ends of the ranking shows that the two annotators share a similar behavior pattern concerning the most and least preferred discriminants. The correlation is slightly weaker in the middle ranking zone, where different preferences or annotation styles can be observed.

To further visualize the effect of the ranking model, we highlighted with color the discriminants which are manually annotated by annotator B under a basic setting without using the ranking models. 75% of these "prominent" discriminants are grouped within the top-25% region of the plot. Without surprise, the model B gives an average relative ranking of 0.18 as oppose to 0.21 with model A. The overall distribution of rankings for manually disambiguated discriminants are shown in Figure 3.

In Table 2, the average treebanking speed of two annotators over multiple annotation sessions is shown. The baseline model ranks the discriminants by the spanning length of the corresponding constituent, and uses the alphabetical order of the rule or lexical type names as tie-breaker. The own-model refers to the annotation sessions which have been carried out by the annotators us-

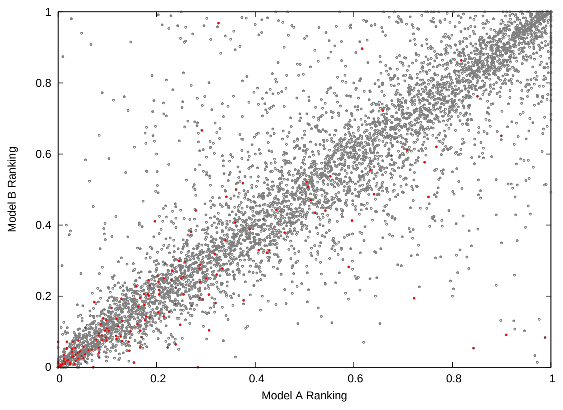


Figure 2: Correlation of discriminant ranks with different models and manual annotation

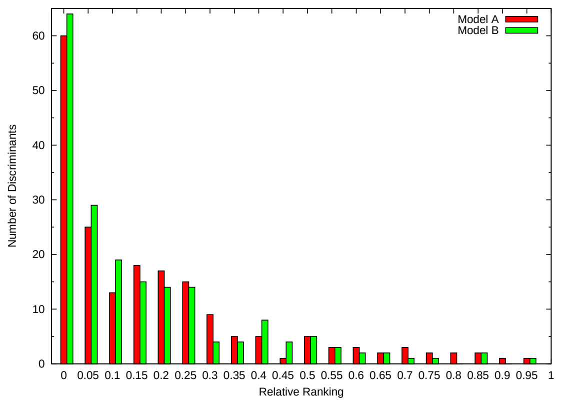


Figure 3: Histogram of rankings given by two models on discriminants manually picked by annotator B

ing their own ranking model. The peer-model refers to the annotation sessions where the annotators use their peer colleague's model. And finally, the joint-model refers to the annotations done by the jointly trained model.

The annotation efficiency was boosted by over 50% with all the discriminant ranking models. The own-model setting achieved best speed. This is probably due to the fact that the model most closely reflects the annotation habit of the annotator. But the advantage over other models is very small.

To measure the inter-annotator agreement, we calculate the Cohen's KAPPA (Carletta, 1996) on the constituents of the derivation trees:

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \qquad (2)$$

Ranking Model	Speed (s/h)	Speed-up (%)
Baseline	61.9	–
Own-model	**96.1**	**55%**
Peer-model	94.6	53%
Joint-model	95.0	53%

Table 2: Average annotation speed with different discriminant ranking models

where $Pr(a)$ is the relative observed agreement between annotators, and $Pr(e)$ is the probability of two annotators agreeing by chance. The calculation of $Pr(a)$ can be done in a similar way to the calculation of PARSEVAL labeled bracketing accuracy, while $Pr(e)$ is estimated by averaging the agreement over a large set of tree pairs randomly sampled from the parse forest. Since the calculation of κ takes into account the agreement occurring by chance, it is a safer (though has the tendency of being overly conservative) measure of agreement.

Ranking Model	Cohen's KAPPA (κ)
Baseline	0.5404
Own-model	**0.5798**
Peer-model	0.5567
Joint-model	0.5536

Table 3: Inter-annotator agreement measured by constituent-level Cohen's KAPPA

The numbers in Table 3 show that the use of discriminant ranking models results in a small improvement to the inter-annotator agreement, with the best agreement achieved by each annotator using the model trained with their own annotation records. These numbers are comforting in that the annotation quality is not damaged by our new way to present the linguistic decisions.

Note that the relatively low inter-annotator agreement in this experiment is due to the fact that we used a dataset which involves non-trivial linguistic phenomena that are on average more difficult than the texts in the WSJ corpus. Another fact is that these annotations were done under time pressure. The annotators are not encouraged to go backwards to check and correct the previous sentences during these sessions. On the entire WSJ, we have recorded a stable and persistently higher agreement level at $\kappa = 0.6$. Given the highly detailed linguistic annotations specified by the grammar (over 260 rules and 800 lexical types), this figure indicates a very substantial agreement between our annotators. Our further investigation has shown that the agreement figure hits the ceiling at around $\kappa = 0.65$. Further training and discussion is not rewarded with sustainable improvement of annotation quality.

Apart from the numerical evaluation, we also interview our annotators for subjective feelings about the various ranking models. There is generally a very positive attitude towards all the ranking models over the baseline. An easily decidable discriminant is usually found within the top-3 with very few exceptions, which leads to a self-noticeable speed-up that confirms our numeric findings. It is also interesting to note that, despite the substantial difference between the statistical models, the difference is hardly noticed by the annotators. And the results only show small variations in both the annotation speed, as well as the inter-annotator agreement.

The annotators also claim that the speed-up is somewhat diminished over the "rejected" sentences, for which none of the candidate trees are acceptable. In such cases, the annotators still have to go through a long sequence of discriminants, and sometimes have to redo the previous steps in fear of the chain-effect of wrong decisions. How to compensate for the psychological insatisfaction of rejecting all analyses while maintaining good annotation speed and quality is a new topic for our future research.

6 Conclusion & Future Work

We propose to use a statistical ranking model to assist the discriminant-based treebank annotation. Our experiment shows that such a model, trained on annotation history, brings a huge efficiency improvement together with slightly improved inter-annotator agreement.

Although the reported experiments were carried out on the specific HPSG treebank, we believe that the proposed ranked discriminant-based annotation method can be applied in annotation tasks concerning different linguistic frameworks, or even different layers of linguistic representation. Apart from the specific features presented in Section 3.3, the model itself does not assume a phrase-structure tree annotation, and the discriminants can take various forms. Assuming a "grammar" produces a number of candidate analyses, the annotators can rely on the ranking model to efficiently pick relevant discriminants, and focus on making linguistically relevant decisions. This is especially suitable for large annotation tasks aiming for parallel rich annotation by multiple annotators, where fully manual annotation is not feasible and high inter-annotator agreement hard to achieve.

The ranking model is based on annotation history and influences the future progress of treebanking. It can be dynamically integrated into the treebank development cycles in which the annotation habit evolves over time. Such a model can also shorten the training period for new annotators, which is an interesting aspect for our future investigation.

From a different point of view, the rankings of the discriminants show annotators' confidence on various ambiguities. The clearly uneven distribution over discriminants can also provide grammar writers with interesting feedback, helping with the improvement of the linguistic analysis. We would also like to integrate confidence measures into the computer-assisted treebank annotation process, which could potentially help annotators make difficult decisions, such as whether to reject all trees for a sentence.

References

[Brants et al.2002] Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41.

[Callmeier2001] Callmeier, Ulrich. 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

[Carletta1996] Carletta, Jean. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254.

[Carter1997] Carter, David. 1997. The treebanker: a tool for supervised training of parsed corpora. In

Proceedings of the ACL Workshop on Computational Environments for Grammar Development and Linguistic Engineering, pages 9–15, Madrid, Spain.

[Copestake et al.2005] Copestake, Ann, Dan Flickinger, Carl J. Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3(4):281–332.

[Flickinger2002] Flickinger, Dan. 2002. On building a more efficient grammar by exploiting types. In Oepen, Stephan, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications.

[Hajič et al.2000] Hajič, Jan, Alena Böhmová, Eva Hajičová, and Barbora Vidová-Hladká. 2000. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In Abeillé, A., editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Amsterdam:Kluwer.

[King et al.2003] King, Tracy H., Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora, held at the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, Budapest, Hungary.

[Kordoni and Zhang2009] Kordoni, Valia and Yi Zhang. 2009. Annotating wall street journal texts using a hand-crafted deep linguistic grammar. In *Proceedings of The Third Linguistic Annotation Workshop (LAW III)*, Singapore.

[Marcus et al.1993] Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

[Oepen et al.2002] Oepen, Stephan, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: motivation and preliminary applications. In *Proceedings of COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, Taipei, Taiwan.

[Oepen2001] Oepen, Stephan. 2001. [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

[Pollard and Sag1994] Pollard, Carl J. and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA.