Using Treebanking Discriminants as Parse Disambiguation Features

Md. Faisal Mahbub Chowdhury[†] **and Yi Zhang**[‡] **and Valia Kordoni**[‡]
[†] Dept of Computational Linguistics, Saarland University
[‡] Dept of Computational Linguistics, Saarland University and DFKI GmbH, Germany
{chowd,yzhang,kordoni}@coli.uni-sb.de

Abstract

This paper presents a novel approach of incorporating fine-grained treebanking decisions made by human annotators as discriminative features for automatic parse disambiguation. To our best knowledge, this is the first work that exploits treebanking decisions for this task. The advantage of this approach is that use of human judgements is made. The paper presents comparative analyses of the performance of discriminative models built using treebanking decisions and state-of-the-art features. We also highlight how differently these features scale when these models are tested on out-of-domain data. We show that, features extracted using treebanking decisions are more efficient, informative and robust compared to traditional features.

1 Introduction

State-of-the-art parse disambiguation models are trained on treebanks, which are either fully hand-annotated or manually disambiguated from the parse forest produced by the parser. While most of the hand-annotated treebanks contain only gold trees, treebanks constructed from parser outputs include both preferred and non-preferred analyses. Some treebanking environments (such as the SRI Cambridge TreeBanker (Carter, 1997) or `[incr tsdb()]` (Oepen, 2001)) even record the treebanking decisions (see section 2) that the annotators take during manual annotation. These treebanking decisions are, usually, stored in the database/log files and used later for dynamic propagation if a newer version of the grammar on the same corpus is available (Oepen et al., 2002). But until now, to our best knowledge, no research has been reported on exploiting these decisions for building a parse disambiguation model.

Previous research has adopted two approaches to use treebanks for disambiguation models. One approach, known as generative, uses only the gold parse trees (Ersan and Charniak, 1995; Charniak, 2000). The other approach, known as discriminative, uses both preferred trees and non-preferred trees (Johnson et al., 1999; Toutanova et al., 2005). In this latter approach, features such as local configurations (i.e., local sub-trees), grandparents, n-grams, etc., are extracted from all the trees and are utilized to build the model. Neither of the approaches considers cognitive aspects of treebanking, i.e. the fine-grained decision-making process of the human annotators.

In this paper, we present our ongoing study of using treebanking decisions for building a parse disambiguation model. We present comparative analyses among the features extracted using treebanking decisions and the state-of-the-art feature types. We highlight how differently these features scale when they are tested on out-of-domain data. Our results demonstrate that features extracted using treebanking decisions are more efficient, informative and robust, despite the total number of these features being much less than that of the traditional feature types.

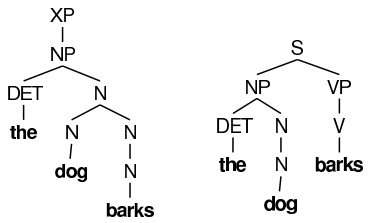
The rest of this paper is organised as follows — section 2 presents some motivation along with definition of treebanking decisions. Section 3 describes the feature extraction templates that have been used for treebanking decisions. Section 4 explains the experimental data, results and analyses. Section 5 concludes the paper with an outline of our future research.

2 Treebanking decisions

One of the defining characteristics of Redwoods-style treebanks[1] (Oepen et al., 2002) is that the candidate trees are constructed automatically by

[1] More details available in http://redwoods.stanford.edu.

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 226–229,
Paris, October 2009.



D1	SUBJH	the dog || barks
D2	HSPEC	the || dog barks
D3	FRAG_NP	the dog barks
D4	HSPEC	the || dog
D5	NOUN_N_CMPND	dog || barks
	...	...
D6	PLUR_NOUN_ORULE	barks
D7	*v_-_le*	barks
D8	*n_-_mc_le*	barks

Figure 1: Example forest and discriminants

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()]` computes 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 only one parse remains. 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, a feature that distinguishes between different parses is called a *discriminant*. For Redwoods-style treebanks, this is usually extracted from the syntactic derivation tree of the Head-driven Phrase Structure Grammar (HPSG) analyses. Figure 1 shows a set of example discriminants based on the two candidate trees.

A choice (acceptance or rejection, either manually annotated or inferred by the system) made on a discriminant is called a *decision*. In the above example, suppose the annotator decides to accept the binary structure *the dog || barks* as a subject-head construction and assigns a value *yes* to discriminant *D1*, the remaining discriminants will also receive inferred values by deduction (*no* for *D2*, *no* for *D3*, *yes* for *D4*, etc). These decisions are stored and used for dynamic evolution of the treebank along with the grammar development.

Treebank decisions (especially those made by annotators) are of particular interest to our study of parse disambiguation. The decisions record the fine-grained human judgements in the manual disambiguation process. This is different from the traditional use of treebanks to build parse selection models, where a marked gold tree is picked from the parse forest without concerning detailed selection steps. Recent study on double annotated treebanks (Kordoni and Zhang, 2009) shows that annotators tend to start with the decisions with the most certainty, and delay the "hard" decisions as much as possible. As the decision process goes, many of the "hard" discriminants will receive an inferred value from the certain decisions. This greedy approach helps to guarantee high inter-annotator agreement. Concerning the statistical parse selection models, the discriminative nature of these treebanking decisions suggests that they are highly effective features, and if properly used, they will contribute to an efficient disambiguation model.

3 Treebanking Decisions as Discriminative Disambiguation Features

We use three types of feature templates for treebanking decisions for feature extraction. We refer to the features extracted using these templates as *TDF* (Treebanking Decision Feature) in the rest of this paper. The feature templates are

T1: *discriminant + lexical types of the yield*

T2: *discriminant + rule(left-child)*[2] *+ rule(right-child)*

T3: *instances of T2 + rule(parent) + rule(siblings)*

*TDF*s of T1, T2 and T3 in combination are referred to as *TDFC* or *TDF*s with context. For example in Figure 1, *instance of T1* for the discriminant *D4* is "HSPEC[3] + *le_type(the)*[4] + *le_type(dog)*"; *instance of T2* is "HSPEC + *rule(*DET*)* + *rule(*N*)* "; and *instance of T3* is "HSPEC + *rule(*DET *)* + *rule(*N*)* + *rule(*S*)* + *rule(*VP*)*".

A *TDF* represents partial information about the right parse tree (as most usual features). But in some way, it also indicates that it was a point of a decision (point of ambiguity with respect to the underlying pre-processing grammar), hence carrying some extra bit of information. *TDF*s allow to

[2] *rule(X)* represents the HPSG rule, applied on X, extracted from the corresponding derivation tree.

[3] HSPEC is the head-specifier rule in HPSG

[4] *le_type(X)* denotes the abstract lexical type of word *X* inside the grammar.

omit certain details inside the features by encoding useful purposes of relationships between lexical types of the words and their distant grandparents without considering nodes in the intermediate levels (allowing some kind of underspecification). In contrast, state-of-the-art feature types contain all the nodes in the corresponding branches of the tree. While they encode ancestor information (through grandparenting), but they ignore siblings. *TDF*s include siblings along with ancestor. Unlike traditional features, which are generated from all possible matches (which is huge) of feature types followed by some frequency cut-offs, the selection of *TDF*s is directly restricted by the small number of treebanking decisions themselves and exhaustive search is not needed. It should be noted that, we do not use treebanking decisions made for the parse forest of one sentence to extract features from the parse forest of another sentence. That is why, the number of *TDF*s is much smaller than that of traditional features. This also ensures that *TDF*s are highly correlated to the corresponding constructions and corresponding sentences from where they are extracted.

4 Experiment

4.1 Data

We use a collection of 8593 English sentences from the LOGON corpus (Oepen et al., 2004) for our experiment. 874 of them are kept as test items and the remaining 7719 items are used for training. The sentences have an average length of 14.68 and average number of 203.26 readings per sentence. The out-of-domain data are a set of 531 English Wikipedia sentences from WeScience corpus (Ytrestøl et al., 2009).

Previous studies (Toutanova et al., 2005; Osborne and Baldridge, 2004) have reported relatively high exact match accuracy with earlier versions of ERG (Flickinger, 2000) on datasets with very short sentences. With much higher structural ambiguities in LOGON and WeScience sentences, the overall disambiguation accuracy drops significantly.

4.2 Experimental setup and evaluation measures

The goal of our experiments is to compare various types of features (with TDF) in terms of efficiency, informativeness, and robustness. To compare among the feature types, we build log-linear training models (Johnson et al., 1999) for parse selection (which is standard for unification-based grammars) for TDFC, local configurations, n-grams and active edges[5]. For each model, we calculate the following evaluation metrics —

- *Exact (match) accuracy*: it is simply the percentage of times that the top-ranked analysis for each test sentences is identical with the gold analysis of the same sentence.
- *5-best (match) accuracy*: it is the percentage of times that the five top-ranked analyses for each of the sentences contain the gold analysis.
- *Feature Hit Count (FHC)*: it is the total number of occurrences of the features (of a particular feature type) inside all the syntactic analyses for all the test sentences. So, for example, if a feature (of a particular feature type) is observed 100 times, then these 100 occurrences are added to the total FHC.
- *Feature Type Hit Count (FTHC)*: it is the total number of *distinct* features (of the corresponding feature type) observed inside the syntactic analyses of all the test sentences.

While exact and 5-best match measures show relative informativeness and robustness of the feature types, *FHC* and *FTHC* provide a more comprehensive picture of relative efficiencies.

4.3 Results and discussion

As we can see in Table 1, local configurations achieve highest accuracy among the traditional feature types. They also use higher number of features (almost *2.7* millions). *TDFC* do better than both n-grams and active edges, even with a lower number of features. Though, local configurations gain more accuracy than *TDFC*, but they do so at a cost of *50* times higher number of features. This indicates that features extracted using treebanking decisions are more informative.

For out-of-domain data (Table 1), there is a big drop of accuracy for local configurations. Active edges and *TDFC* also have some accuracy drop. Surprisingly, n-grams do better with our out-of-domain data than in-domain, but still that accuracy is close to that of *TDFC*. Note that n-grams have *8* times higher number of features than *TDFC*. Hence, according to these results, *TDFC* are more robust, for out-of-domain data, than local configurations and active edges, and almost as good as n-grams.

[5] Active edges correspond to the branches (i.e. one daughter in turn) of the local sub-trees.

Feature template	Total features	5-best accuracy *(in-domain)*	5-best accuracy *(out-of-domain)*	Exact accuracy *(in-domain)*	Exact accuracy *(out-of-domain)*
n-gram	438,844	68.19%	62.71%	41.30%	42.37%
local configuration	2,735,486	75.51%	64.22%	50.69%	44.44%
active edges	89,807	68.99%	61.77%	41.88%	39.92%
TDFC	53,362	70.94%	62.71%	43.59%	41.05%

Table 1: Accuracies obtained on both in-domain and out-of-domain data using n-grams (n=4), local configurations (with grandparenting level 3), active edges and *TDFC*.

Feature template	FHC	FTHC	Active features
n-gram	18,245,558	32,425	7.39%
local config.	62,060,610	357,150	13.06%
active edges	22,902,404	27,540	30.67%
TDFC	21,719,698	17,818	33.39%

Table 2: *FHC* and *FTHC* calculated for in-domain data.

The most important aspect of *TDFC* is that they are more efficient than their traditional counterparts (Table 2). They have significantly higher number of active features ($\frac{FTHC}{TotalFeature\#}$) than n-grams and local configurations.

5 Future work

The results of the experiments described in this paper indicate a good prospect for utilizing treebanking decisions, although, we think that the types of feature templates that we are using for them are not yet fully conveying cognitive knowledge of the annotators, in which we are specifically interested in. For instance, we expect to model human disambiguation process more accurately by focusing only on human annotators' decisions (instead of only inferred decisions). Such a model will not only improve the performance of the parsing system at hand, but can also be applied interactively in treebanking projects to achieve better annotation speed (e.g., by ranking the promising discriminants higher to help annotators make correct decisions). Future experiments will also investigate whether any pattern of discriminant selection by the humans can be learnt from these decisions.

References

David Carter. 1997. The treebanker: A tool for supervised training of parsed corpora. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, Madrid, Spain.

Eugene Charniak. 2000. A maximum entropy-based parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of Association for Computational Linguistics (NAACL 2000)*, pages 132–139, Seattle, USA.

Murat Ersan and Eugene Charniak. 1995. A statistical syntactic disambiguation program and what it learns. pages 146–159.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. 6(1):15–28.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic unifcation-based grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, pages 535–541, Maryland, USA.

Valia Kordoni 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.

Stephan Oepen, 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.

Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. 2004. Som å kappete med trollet? towards mrs-based norwegian-english machine translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 11–20, MD, USA.

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

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *HLT-NAACL 2004: Main Proceedings*, pages 89–96, Boston, USA.

Kristina Toutanova, Christoper D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, 3(1):83–105.

Gisle Ytrestøl, Stephan Oepen, and Daniel Flickinger. 2009. Extracting and annotating wikipedia sub-domains. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, pages 185–197, Groningen, the Netherlands.