Using Large-scale Parser Output to Guide Grammar Development

Ascander Dost
Powerset, a Microsoft company
adost@microsoft.com

Tracy Holloway King
Powerset, a Microsoft company
Tracy.King@microsoft.com

Abstract

This paper reports on guiding parser development by extracting information from output of a large-scale parser applied to Wikipedia documents. Data-driven parser improvement is especially important for applications where the corpus may differ from that originally used to develop the core grammar and where efficiency concerns affect whether a new construction should be added, or existing analyses modified. The large size of the corpus in question also brings scalability concerns to the foreground.

1 Introduction

Initial development of rule-based parsers[1] is often guided by the grammar writer's knowledge of the language and test suites that cover the "core" linguistic phenomena of the language (Nerbonne et al., 1988; Cooper et al., 1996; Lehmann et al., 1996). Once the basic grammar is implemented, including an appropriate lexicon, the direction of grammar development becomes less clear. Integration of a grammar in a particular application and the use of a particular corpus can guide grammar development: the corpus and application will require the implementation of specific constructions and lexical items, as well as the reevaluation of existing analyses. To streamline this sort of output-driven development, tools to examine parser output over large corpora are necessary, and as corpus size increases, the efficiency and scalability of those tools become crucial concerns. Some immediate relevant questions for the grammar writer include:

- What constructions and lexical items need to be added for the application and corpus in question?
- For any potential new construction or lexical item, is it worth adding, or would it be better to fall back to robust techniques?
- For existing analyses, are they applying correctly, or do they need to be restricted, or even removed?

In the remainder of this section, we briefly discuss some existing techniques for guiding large-scale grammar development and then introduce the grammar being developed and the tool we use in examining the grammar's output. The remainder of the paper discusses development of lexical resources and grammar rules, how overall progress is tracked, and how analysis of the grammar output can help development in other natural language components.

1.1 Current Techniques

There are several techniques currently being used by grammar engineers to guide large-scale grammar development, including error mining to detect gaps in grammar coverage, querying tools for gold standard treebanks to determine frequency of linguistic phenomena, and tools for querying parser output to determine how linguistic phenomena were analyzed in practice.

An error mining technique presented by van Noord (2004) (henceforth: the van Noord Tool) can reveal gaps in grammar coverage by comparing the frequency of arbitrary n-grams of words in unsuccessfully parsed sentences with the same n-grams in unproblematic sentences, for large unannotated corpora.[2] A parser can be run over new text, and a comparison of the in-domain and

[1] The techniques discussed here may also be relevant to purely machine-learned parsers and are certainly applicable to hybrid parsers.

[2] The suffix array error mining software is available at: http://www.let.rug.nl/~vannoord/SuffixArrays.tgz

Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks, ACL-IJCNLP 2009, pages 63–70,
Suntec, Singapore, 6 August 2009.

out-of-domain sentences can determine, for instance, that the grammar cannot parse adjective-noun hyphenation correctly (e.g. *an electrical-switch cover*). A different technique for error mining that uses discriminative treebanking is described in (Baldwin et al., 2005). This technique aims at determining issues with lexical coverage, grammatical (rule) coverage, ungrammaticality within the corpus (e.g. misspelled words), and extragrammaticality within the corpus (e.g. bulleted lists).

A second approach involves querying gold-standard treebanks such as the Penn Treebank (Marcus et al., 1994) and Tiger Treebank (Brants et al., 2004) to determine the frequency of certain phenomena. For example, Tiger Search (Lezius, 2002) can be used to list and frequency-sort stacked prepositions (e.g. *up to the door*) or temporal noun/adverbs after prepositions (e.g. *by now*). The search tools over these treebanks allow for complex searches involving specification of lexical items, parts of speech, and tree configurations (see (Mírovský, 2008) for discussion of query requirements for searching tree and dependency banks).

The third approach we discuss here differs from querying gold-standard treebanks in that corpora of actual parser output are queried to examine how constructions are analyzed by the grammar. For example, Bouma and Kloosterman (2002) use XQuery (an XML query language) to mine parse results stored as XML data.[3] It is this sort of examination of parser output that is the focus of the present paper, and specific examples of our experiences follow in Section 2.2.

Use of such tools has proven vital to the development of large-scale grammars. Based on our experiences with them, we began extensively using a tool called Oceanography (Waterman, 2009) to search parser output for very large (approximately 125 million sentence) parse runs stored on a distributed file system. Oceanography queries the parser output and returns counts of specific constructions or properties, as well as the example sentences they were extracted from. In the subsequent sections we discuss how this tool (in conjunction with existing ones like the van Noord Tool and Tiger Search) has enhanced grammar development for an English-language Lexical-Functional Grammar used for a semantic search application over Wikipedia.

[3] See also (Bouma and Kloosterman, 2007) for further discussion of this technique.

1.2 The Grammar and its Role

The grammar being developed is a Lexical-Functional Grammar (LFG (Dalrymple, 2001)) that is part of the ParGram parallel grammar project (Butt et al., 1999; Butt et al., 2002). It runs on the XLE system (Crouch et al., 2009) and produces c(onstituent)-structures which are trees and f(unctional)-structures which are attribute value matrices recording grammatical functions and other syntactic features such as tense and number, as well as debugging features such as the source of lexical items (e.g. from a named entity finder, the morphology, or the guesser). There is a base grammar which covers the constructions found in standard written English, as well as three overlay grammars: one for parsing Wikipedia sentences, one for parsing Wikipedia headers, and one for parsing queries (sentential, phrasal, and keyword).

The grammar is being used by Powerset (a Microsoft company) in a semantic consumer-search reference vertical which allows people to search Wikipedia using natural language queries as well as traditional keyword queries. The system uses a pipeline architecture which includes: text extraction, sentence breaking, named entity detection, parsing (tokenization, morphological analysis, c-structure, f-structure, ranking), semantic analysis, and indexing of selected semantic facts (see Figure 1). A similar pipeline is used on the query side except that the resulting semantic analysis is turned into a query execution language which is used to query the index.

text extraction	*script*
sentence breaker	*finite state*
named entity detection	*MaxEnt model*
LFG grammars	
tokenizer	*finite state*
morphology	*finite state*
grammar	*XLE: parser*
ranking	*MaxEnt model*
semantics	*XLE: XFR*

Figure 1: NL Pipeline Components

The core idea behind using a deep parser in the pipeline in conjunction with the semantic rules is to localize role information as to who did what to whom (i.e. undo long-distance dependencies and

locate heads of arguments), to abstract away from choice of particular lexical items (i.e. lemmatization and detection of synonyms), and generally provide a more normalized representation of the natural language string to improve both precision and recall.

1.3 Oceanography

As a byproduct of the indexing pipeline, all of the syntactic and semantic structures are stored for later inspection as part of failure analysis.[4] The files containing these structures are distributed over several machines since ∼125 million sentences are parsed for the analysis of Wikipedia.

For any given syntactic or semantic structure, the XLE ordered rewrite system (XFR; (Crouch et al., 2009)) can be used to extract information that is of interest to the grammar engineer, by way of "rules" or statements in the XFR language. As the XFR ordered rewrite system is also used for the semantics rules that turn f-structures into semantic representations, the notation is familiar to the grammar writers and is already designed for manipulating the syntactic f-structures.

However, the mechanics of accessing each file on each machine and then assembling the results is prohibitively complicated without a tool that provides a simple interface to the system. Oceanography was designed to take a single specification file stating:

- which data to examine (which corpus version; full Wikipedia build or fixed 10,000 document set);
- the XFR rules to be applied;
- what extracted data to count and report back.

Many concrete examples of Oceanography runs will be discussed below. The basic idea is to use the XFR rules to specify searches over lexical items, features, and constructions in a way that is similar to that of Tiger Search and other facilities. The Oceanography machinery enables these searches over massive data and helps in compiling the results for the grammar engineer to inspect. We believe that similar approaches would be feasible to implement in other grammar development environments and, in fact, for some grammar outputs and applications, existing tools such as Tiger Search would be sufficient. By providing examples where such searches have aided our grammar development, we hope to encourage other grammar engineers to similarly extend their efforts to use easy access to massive data to drive their work.

[4]The index is self-contained and does not need to reference the semantic, much less the earlier syntactic, structures as part of the search application.

2 Grammar Development

The ParGram English LFG grammar has been developed over many years. However, the focus of development was on newspaper text and technical manuals, although some adaptation was done for new domains (King and Maxwell, 2007). When moving to the Wikipedia domain, many new constructions and lexical items were encountered (see (Baldwin et al., 2005) for a similar experience with the BNC) and, at the same time, the requirements on parsing efficiency increased.

2.1 Lexical Development

When first parsing a new corpus, the grammar encounters new words that were previously unknown to the morphology. The morphology falls back to a guesser that uses regular expressions to guess the part of speech and other features associated with an unknown form. For example, a novel word ending in *s* might be a plural noun. The grammar records a feature LEX-SOURCE with the value *guesser* for all guessed words. Oceanography was used to extract all guessed forms and their parts of speech. In many cases, the guesser had correctly identified the word's part of speech. However, words that occurred frequently were added to the morphology to avoid the possibility that they would be incorrectly guessed as a different part of speech. The fact that Oceanography was able to identify not just the word, but its posited part of speech and frequency in the corpus greatly sped lexical development.

Incorrect guessing of verbs was of particular concern to the grammar writers, as misidentification of verbs was almost always accompanied by a bad parse. In addition, subcategorization frames for guessed verbs were guessed as either transitive or intransitive, which often proved to be incorrect. As such, the guessed verbs extracted using Oceanography were hand curated: true verbs were added to the morphology and their subcategorization frames to the lexicon. Due to the high rate of error with guessed verbs, once the correctly guessed verbs were added to the morphology, this

option was removed from the guesser.[5]

Overall, ∼4200 new stems were added to the already substantial morphology, with correct inflection. Approximately ∼1300 of these were verbs. The decision to eliminate verbs as possible guessed parts of speech was directly motivated by data extracted using Oceanography.

Since the guesser works with regular expressions (e.g. lowercase letters + *s* form plural nouns), it is possible to encounter forms in the corpus that neither the morphology nor the guesser recognize. The grammar will fragment on these sentences, creating well-formed f-structure chunks but no single spanning parse, and the unrecognized forms will be recorded as TOKENs (Riezler et al., 2002). An Oceanography run extracting all TOKENs resulted in the addition of several new patterns to the guesser as well as the addition of some of the frequent forms to the morphology. For example, sequences of all upper case letters followed by a hyphen and then by a sequence of digits were added for forms like *AK-47*, *F-22*, and *V-1*.

The guesser and TOKENs Oceanography runs look for general problems with the morphology and lexicon, and can be run for every new corpus. More specific jobs are run when evaluating whether to implement a new analysis, or when evaluating whether a current analysis is functioning properly. For example, use of the van Noord tool indicated that the grammar had problems with certain less common multiword prepositions (e.g. *pursuant to*, *in contrast with*). Once these multiword prepositions were added, the question then arose as to whether more common prepositions should be multiwords when stacked (e.g. *up to*, *along with*). An Oceanography run was performed to extract all occurrences of stacked prepositions from the corpus. Their frequency was tallied in both the stacked formations and when used as simple prepositions. With this information, we determined which stacked configurations to add to the lexicon as multiword prepositions, while maintaining preposition stacking for less common combinations.

2.2 Grammar Rule Development

In addition to using Oceanography to help develop the morphology and lexicon, it has also proven extremely useful in grammar rule development. In general, the issue is not in finding constructions which the grammar does not cover correctly: a quick investigation of sentences which fragment can provide these and issues are identified and reported by the semantics which uses the syntax output as its input. Furthermore, the van Noord tool can be used to effectively identify gaps in grammar rule coverage.

Rather, the more pressing issues include whether it is worthwhile adding a construction, which possible solution to pick (when it is worthwhile), and whether an existing solution is applying correctly and efficiently. Being able to look at the occurrence of a construction over large amounts of data can help with all of these issues, especially when combined with searching over gold standard treebanks such as the Penn Treebank.

Determining which constructions to examine using Oceanography is often the result of failure analysis findings on components outside the grammar itself, but that build on the grammar's output later in the natural language processing pipeline. The point we wish to emphasize here is that the grammar engineer's effectiveness can greatly benefit from being able to take a set of problematic data gathered from massive parser output and determine from it that a particular construction merits its closer scrutiny.

2.2.1 *When* relative/subordinate clauses

An observation that subordinate clauses containing *when* (e.g. *Mary laughed when Ed tripped.*) were sometimes misanalyzed as relative clauses attaching to a noun (e.g. *the time when Ed tripped*) prompted a more directed analysis of whether *when* relative clauses should be allowed to attach to nouns that were not time-related expressions (e.g. *time, year, day*). An Oceanography run was performed to extract all *when* relative clauses, the modified nominal head, and the sentence containing the construction. A frequency-sorted list of nouns taking *when* relative clause modifiers helped to direct hand-examination of *when* relative clauses for accuracy of the analysis. This yielded some correct analyses:

(1) There are **times** [when a Bigfoot sighting or footprint is a hoax].

More importantly, however, the search revealed many incorrect analyses of *when* subordinate

[5] It is simple to turn the guessed verbs back on in order to run the same Oceanography experiment with a new corpus.

clauses as relative clauses:

(2) He gets the last **laugh** [when he tows away his boss' car as well as everyone else's].

By extracting all *when* relative clauses, and their head nouns, it was determined that the construction was generally only correct for a small class of time expression nominals. Comparatively, *when* relative clause modification of other nominals was rarely correct. The grammar was modified to dispref relative clause analyses of *when* clauses unless the head noun was an expression of time. As a result, the overall quality of parses for all sentences containing *when* subordinate clauses was improved.

2.2.2 Relative clauses modifying gerunds

Another example of an issue with the accuracy of a grammatical analysis concerns gerund nouns modified by relative clauses without an overt relative pronoun (e.g. *the singing we liked*). It was observed that many strings were incorrectly analyzed as a gerund and reduced relative clause modifier:

(3) She lost all of her powers, **including** [her sonic screams].

Again, a frequency sorted list of gerunds modified by reduced relative clauses helped to guide hand inspection of the instances of this construction. By extracting all of the gerunds with reduced relative clause modifiers, it was possible to see which gerunds were appearing in this construction (e.g. *including* occurred alarmingly frequently) and how rarely the overall analysis was correct. As a result of the data analysis, such relative clause modifiers are now dispreferred in the grammar and certain verbs (e.g. *include*) are additionally dispreferred as gerunds in general. Note that this type of failure analysis is not possible with a tool (such as the van Noord tool) that only points out gaps in grammar coverage.

2.2.3 Noun-noun compounds

As part of the semantic search application, argument-relation triples are extracted from the corpus and presented to the user as a form of summary over what Wikipedia knows about a particular entity. These are referred to as *Factz*. For example, a search on *Noam Chomsky* will find Factz triples as in Figure 2. Such an application highlights parse problems, since the predicate-argument relations displayed are ultimately extracted from the syntactic parses themselves.

One class of problem arises when forms which are ambiguous between nominal and verbal analyses are erroneously analyzed as verbs and hence show up as Factz relations. This is particularly troublesome when the putative verb is part of a noun-noun compound (e.g. *ice cream, hot dog*) and the verb form is comparatively rare. A list of potentially problematic noun-noun compounds was extracted by using an independent part of speech tagger over the sentences that generated the Factz triples. If the relation in the triple was tagged as a noun and was not a deverbal noun (e.g. *destruction, writing*), then the first argument of the triple and the relation were tagged as potentially problematic noun-noun compounds. Oceanography was then used to determine the relative frequency of whether the word pairs were analyzed as noun-noun compounds, verb-argument relations, or independent nouns and verbs.

This distributional information, in conjunction with information about known noun-noun compounds in WordNet (Fellbaum, 1998), is being used to extract a set of $\sim$100,000 noun-noun compounds whose analysis is extremely strongly preferred by the grammar. Currently, these are constrained via c-structure optimality marks[6] but they may eventually be allowed only as noun-noun compounds if the list proves reliable enough.

3 Tracking Grammar Progress

The grammar is used as part of a larger application which is actively being developed and which is regularly updated. As such, new versions of the grammar are regularly released. Each release includes a detailed list of improvements and bug fixes, as well as requirements on other components of the system (e.g. the grammar may require a specific version of the XLE parser or of the morphology). It is extremely important to be able to confirm that the changes to the grammar are in place and are functioning as expected when used in the pipeline. Some changes can be confirmed by browsing documents, finding a sentence likely to contain the relevant lexical item or construction, and then inspecting the syntactic structures for that

[6] See (Frank et al., 2001) and (Crouch et al., 2009) on the use of Optimality Theory marks within XLE. C-structure optimality marks apply preferences to the context free backbone before any constraints supplied by the f-structure annotations are applied. This means that the noun-noun compounds will be the only analysis possible if any tree can be constructed with them.

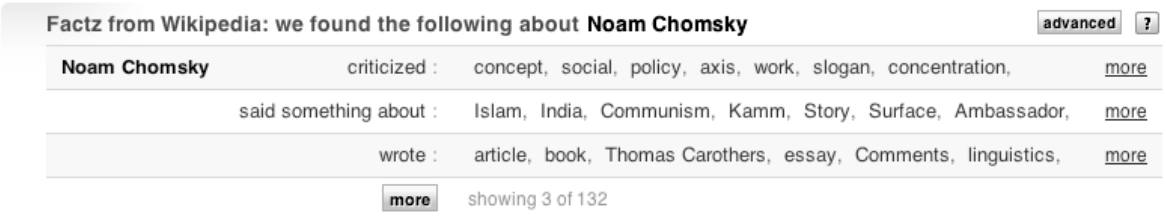

Figure 2: Example Factz

document.

3.1 Confirming Grammar Changes

However, some changes are more complicated to confirm either because it is hard to determine from a sentence whether the grammar change would apply or because the change is more frequency related. For these types of changes, Oceanography runs can detect whether a rare change occurred at all, alleviating the need to search through documents by hand. For example, to determine whether the currency symbols are being correctly treated by the grammar, especially the ones that are not standard ASCII (e.g. the euro and yen symbols), two simple XFR rules can be written: one that looks for the relevant c-structure leaf node and counts up which symbols occur under this node and one that looks for the known list of currency symbols in the f-structure and counts up what part-of-speech they were analyzed as.

To detect whether frequency related changes to the grammar are behaving as expected, two Oceanography runs can be compared, one with the older grammar and one with the newer one. For example, to determine whether relative clauses headed by *when* were dispreferred relative to subordinate clauses, the number of such relative clauses and such subordinate clauses were counted in two successive runs; the relative occurrence of the types confirmed that the preference mechanism was working correctly. In addition, a quick examination of sentences containing each type showed that the change was not over-applying (e.g. incorrectly analyzing *when* relative clauses as subordinate clauses).

3.2 General Grammar Checking

In addition to Oceanography runs done to check on specific changes to the grammar, a core set of XFR rules extracts all of the features from the f-structure and counts them. The resulting statistics of features and counts are computed for each major release and compared to that of the previous release. This provides a list of new features which subsequent components must be alerted to (e.g. a feature added to indicate what type of punctuation surrounded a parenthetical). It also provides a quick check of whether some feature is no longer occurring with the same frequency. In some cases this is expected; once many guessed forms were added to the lexicon, the feature indicating that the guesser had applied dropped sharply. However, unexpected steep variations from previous runs can be investigated to make sure that rules were not inadvertently removed from the grammar, and that rules added to the grammar are functioning correctly.

4 Using Grammar Output to Develop Other Components

In addition to being used in development of the grammar itself, examination of the grammar output can be useful for engineering efforts on other components. In addition to the examples cited above concerning the development of the morphology used by the grammar, we discuss one simple example here. The sentence breaker used in the pipeline is designed for high precision; it only breaks sentences when it is sure that there is a sentence break. To make up for breaks that may have been missed, the grammar contains a rule that allows multiple sentences to be parsed as a single string. The resulting f-structure has the final sentence's f-structure as the value of a feature, LAST, and the remainder as the value of a feature, REST. The grammar iteratively parses multiple sentences into these LAST-REST structures. Because the feature LAST is only instantiated when parsing multiple sentences, input strings whose parses contained a LAST component could be extracted to determine whether the sentence breaker's behavior should be changed. An example of two sentences which were not broken is:

(4) The current air staff includes former CNN Headline News gal Holly Firfer in the mornings with co-host Orff. Mid-days is Mara Davis, who does a theme lunch hour.

The relatively short unknown word *Orff* before the period makes it unclear whether this is an abbreviation or not. Based on the Oceanography analysis, the number of unbroken sentences which received analyses was roughly halved and one bug concerning footnote markers was discovered and fixed.

5 Conclusion

Large-scale grammars are increasingly being used in applications. In order to maximize their effectiveness in terms of coverage, accuracy, and efficiency for a given application, it is increasingly important to examine the behavior of the grammar on the relevant corpus and in the relevant application.

Having good tools makes the grammar engineer's task of massive data driven grammar development significantly easier. In this paper we have discussed how such a tool, which can apply search patterns over the syntactic (and semantic) representations of Wikipedia, is being used in a semantic search research vertical. When used in conjunction with existing tools for detecting gaps in parser coverage (e.g. the van Noord tool), Oceanography greatly aids in the evaluation of existing linguistic analyses from the parser. In addition, oceanography provides vital information to determining whether or not to implement coverage for a particular construction, based on efficiency requirements. Thus, the grammar writer has a suite of tools available to address the questions raised in the introduction of this paper: what gaps exist in parser coverage, how to best address those gaps, and whether existing analyses are functioning appropriately. We hope that our experiences encourage other grammar engineers to use similar techniques in their grammar development efforts.

Acknowledgments

We would like to thank Scott Waterman for creating Oceanography and adapting it to our needs.

References

Timothy Baldwin, John Beavers, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2005. Beauty and the beast: What running a broad-coverage precision grammar over thee bnc taught us about the grammar — and the corpus. In Stephan Kepser and Marga Reis, editors, *Linguistic Evidence: Empirical, Theoretical, and Computational Perspectives*, pages 49–70. Mouton de Gruyter, Berlin.

Gosse Bouma and Geert Kloosterman. 2002. Querying dependency treebanks in XML. In *Proceedings of the Third international conference on Language Resources and Evaluation (LREC)*, Gran Canaria.

Gosse Bouma and Geert Kloosterman. 2007. Mining syntactically annotated corpora using XQuery. In *Proceedings of the Linguistic Annotation Workshop*, Prague, June. ACL.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2:597–620.

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications.

Miram Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *COLING2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Robin Cooper, Dick Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. 1996. Using the framework. FraCas: A Framework for Computational Semantics (LRE 62-051).

Dick Crouch, Mary Dalrymple, Ronald Kaplan, Tracy Holloway King, John T. Maxwell III, and Paula Newman. 2009. XLE Documentation. Online.

Mary Dalrymple. 2001. *Lexical Functional Grammar*. Syntax and Semantics. Academic Press.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

Anette Frank, Tracy Holloway King, Jonas Kuhn, and John T. Maxwell III. 2001. Optimality theory style constraint ranking in large-scale LFG grammars. In Peter Sells, editor, *Formal and Empirical Issues in Optimality Theoretic Syntax*, pages 367–397. CSLI Publications.

Tracy Holloway King and John T. Maxwell, III. 2007. Overlay mechanisms for multi-level deep processing applications. In *Proceedings of the Grammar Engineering Across Frameworks (GEAF07) Workshop*. CSLI Publications.

Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. 1996. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING 1996*.

Wolfgang Lezius. 2002. *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora (in German)*. Ph.D. thesis, IMS, University of Stuttgart Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS). volume 8, number 4.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

Jiří Mírovský. 2008. PDT 2.0 requirements on a query language. In *Proceedings of ACL-08: HLT*, pages 37–45. Association for Computational Linguistics.

John Nerbonne, Dan Flickinger, and Tom Wasow. 1988. The HP Labs natural language evaluation tool. In *Proceedings of the Workshop on Evaluation of Natural Language Processing Systems*.

Stefan Riezler, Tracy Holloway King, Ronald Kaplan, Dick Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the ACL*.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *Proceedings of ACL*.

Scott A. Waterman. 2009. Distributed parse mining. In *Proceedings of the NAACL Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.