PARSER EVALUATION USING TEXTUAL ENTAILMENTS

by

Önder Eker

B.S., Computer Engineering, Boğaziçi University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2009

ACKNOWLEDGEMENTS

I would like to thank Assoc. Prof. Tunga Güngör and Assist. Prof. Deniz Yüret for their guidance.

I would like to thank Dr. Ali Vahit Şahiner, Assist. Prof. Alper Şen and Dr. Tamer Şıkoğlu for kindly accepting to be committee members.

I would like to thank my family for their love, support and encouragement.

The author was supported by TÜBİTAK MS Fellowship BİDEB-2228.

ABSTRACT

PARSER EVALUATION USING TEXTUAL ENTAILMENTS

Syntactic parsing is a basic problem in natural language processing. It can be defined as assigning a structure to a sentence. Two prevalent approaches to parsing are phrase-structure parsing and dependency parsing. A related problem is parser evaluation. This thesis proposes Parser Evaluation using Textual Entailments as a dependency-based evaluation where a parse is represented as a list of simple sentences, similar to the Recognizing Textual Entailments task. Each entailment focuses on one relation. A priori training of annotators is not required. A program generates entailments from a dependency parse. Phrase-structure parses are converted to dependency parses to generate entailments. Additional entailments are generated for phrase-structure coordinations. Experiments are carried out with a function-tagger. Parsers are evaluated on the set of entailments generated from the Penn Treebank WSJ and Brown test sections. A phrase-structure parser obtained the highest score.

ÖZET

METİNSEL GEREKTİRİMLER İLE AYRIŞTIRICI DEĞERLENDİRMESİ

Sözdizimsel ayrıştırma doğal dil işlemede temel bir problemdir. Cümleye bir yapı atamak olarak tanımlanabilir. En yaygın iki ayrıştırma, öbek yapısı ayrıştırma ve bağımsallık ayrıştırmasıdır. İlgili bir konu ayrıştırıcı değerlendirmesidir. Bu tez, ayrıştırmanın Metinsel Gerektirimleri Tanıma görevinde olduğu gibi bir dizi basit cümle ile ifade edildiği bağımsallık tabanlı bir değerlendirme olan Metinsel Gerektirimler ile Ayrıştırıcı Değerlendirmesini önermektedir. Her gerektirim bir bağlantıya odaklanmaktadır. Yorumcuların önceden eğitilmesine gerek yoktur. Bir program bağımsallık ayrıştırmasından gerektirimleri üretmektedir. Öbek yapısı ayrıştırmaları gerektirim üretmek için bağımsallık ayrıştırmasına çevrilmektedir. Öbek yapısı eşgüdümlerinden ek gerektirimler üretilmektedir. Bir işlev etiketçi ile deneyler yapılmıştır. Ayrıştırıcılar Penn Treebank WSJ ve Brown test kısımlarından üretilen gerektirim kümesi üzerinde değerlendirilmiştir. Bir öbek yapısı ayrıştırıcı en yüksek puanı almıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

c	Conjunct
D	Dependency parse tree
e	Entailment
E	List of entailments
h	Head
l	Lowest common ancestor
P	Phrase-structure parse tree
s	Subtree
S	Sentence
T	Parse
w	Word
W	List of words

CCG	Combinatory Categorial Grammar
CKY	Cocke-Kasami-Younger
CoNLL	Conference on Computational Natural Language Learning
GR	Grammatical Relation
IE	Information Extraction
IR	Information Retrieval
LAS	Labeled Attachment Score
LFG	Lexical Functional Grammar
LHS	Left Hand Side
MST	Maximum Spanning Tree
MT	Machine Translation
PCFG	Probabilistic Context-Free Grammar
PETE	Parser Evaluation using Textual Entailments
POS	Part-of-speech
PRD	Predicate
QA	Question Answering

RHS	Right Hand Side
RTE	Recognizing Textual Entailments
SBJ	Subject
SUM	Summarization
TMP	Temporal
UAS	Unlabeled Attachment Score
WSJ	Wall Street Journal

1. INTRODUCTION

Parsing is a basic problem in natural language processing. Broadly it can be defined as assigning a structure to a sentence [1]. Different formalisms have been proposed to represent the sentence structure. However, the common goal is to provide semantic analysis for downstream applications.

Recently, data-driven methods in syntactic parsing have gained prominence. Instead of manually specifying grammar rules, they rely on a treebank of syntactically annotated sentences. The most widely used treebank is the Penn Treebank.

The Penn Treebank is annotated with part-of-speech (POS) tags and syntactic bracketings [2]. A two-step approach is followed in both tagging processes. A program assigns POS tags automatically. Annotators correct the assigned tags. An annotator becomes fluent after a training period of one month and can annotate 3000 words per hour. Similarly, a deterministic parser provides the initial parse. Annotators then correct or combine the syntactic groups. It may take two months to become fluent in this task. An annotator does syntactic annotation at a rate of approximately 500 words per hour.

Creating a treebank is a non-trivial task. Certain conventions should be set forth. Any disagreements arising during the annotation process should be resolved. In the end, the decisions underlying a treebank amount to a significant volume. For example, bracketing manual for the Penn Treebank [3] has more than 300 pages. Creating this manual has taken eight months of ten-hour a week effort [4].

Given the complexity of the linguistic representation, there are some problems in the annotated data. Different annotations for two similar text segments that could have been identically annotated cause inconsistencies. They may be caused by genuine annotator errors or by permissible alternatives as listed in the manual [3]. Since inconsistencies are noisy data points, they are problematic for the training and evaluation

of statistical parsers. In training, all data cannot be used effectively. Evaluation on noisy data is not meaningful and may give incorrect assessments and rankings.

Dickinson and Meurers have developed a method to detect inconsistencies in syntactic annotations [5]. They consider identical strings having different syntactic category labels. Using a similar method, Yuret estimates that the Penn Treebank has at least 15 per cent inconsistency [6].

Besides giving rise to annotation difficulties, the linguistic representation of the Penn Treebank may not be the most appropriate for semantic understanding [1]. Performance on recovering dependencies such as subjects, objects may be more relevant. Dependency-based evaluation considers the precision and recall on grammatical relations. See Chapter 2 for a review of these methods.

Existing dependency-based evaluation schemes have some problems as well [7]. Notation is sometimes unclear and documentation inadequate. Conversion between dependency formats may be problematic, harming the fairness of evaluation. There is also some discussion on the choice of grammatical relations and their arguments.

This thesis proposes Parser Evaluation using Textual Entailments (PETE). PETE is a dependency-based evaluation scheme where parser output is represented as a list of simple sentences called textual entailments. Each entailment focuses on one relation. The relation words are always included. Other words may be added to satisfy grammaticality. Dependency type is not explicitly specified. Coarse-grained dependency type information is implicit in the entailment type and the ordering of words.

The proposed method has advantages in annotation and evaluation. Training of annotators is not required. Many people can participate in the annotation via the Web. Inter-annotator agreement is expected to be higher. Annotation is expected to take less time, to be cheaper and more accurate. In this thesis only evaluation is investigated. Test set of entailments are generated from the Penn Treebank. Phrase-structure and dependency parsers are evaluated. Evaluation is relevant to semantic understanding

since each entailment is centered on a semantically meaningful relation.

The thesis is organized as follows. Chapter 2 describes two direct influences to this thesis, Recognizing Textual Entailments and dependency-based parser evaluation. Chapter 3 describes statistical parsing, input formats, evaluation metrics and introduces the parsers evaluated in this thesis. Chapter 4 explains how entailments are generated. Chapter 5 gives evaluation results. Chapter 6 concludes the thesis with remarks on future directions.

2. BACKGROUND

PETE is a dependency-based parser evaluation. Instead of specifying a dependency type, the relation words are represented in a simple sentence, similar to a hypothesis sentence in Recognizing Textual Entailments. This chapter describes these two direct influences to PETE.

2.1. Recognizing Textual Entailments

Recognizing Textual Entailments (RTE) is a series of challenges carried out in 2005 [8], 2006 [9] and 2007 [10]. The task is to recognize whether a hypothesis text can be reasonably inferred from the source text. RTE proposes a generic framework for researchers working in different areas such as machine translation (MT) evaluation, question answering (QA) and information extraction (IE).

Source text usually contains multiple sentences, while the hypothesis text is a single sentence. Yes and no answers are equally distributed. Considering the various potential applications, data have been collected from different sources. For example, a gold standard human translation is taken as the source text and an automatic translation is taken as the hypothesis text. There are approximately 1500 source–hypothesis text pairs in development and test data. Some examples are given in Table 2.1.

The submitted systems use a range of resources and methods [10]. WordNet and the Web are frequently used. Some systems implement logical inference. Some systems carry out anaphora resolution and named entity recognition. Machine learning with lexical and syntactic features have proven useful. The highest score obtained in the third challenge is around 80 per cent.

A relevant observation is that syntax by itself has less than 50 per cent applicability for RTE [11]. In this study human annotators assume that gold standard syntactic parse and a general thesaurus are available and try to find out the upper bound using

Table 2.1. Examples of text-hypothesis pairs from RTE-3 [10]

Task	**Text**	**Hypothesis**	**Entailment**
IE	At the same time the Italian digital rights group, Electronic Frontiers Italy, has asked the nation's government to investigate Sony over its use of anti-piracy software.	Italy's government investigates Sony.	No
IR	Between March and June, scientific observers say, up to 300,000 seals are killed. In Canada, seal-hunting means jobs, but opponents say it is vicious and endangers the species, also threatened by global warming.	Hunting endangers seal species.	Yes
QA	Aeschylus is often called the father of Greek tragedy; he wrote the earliest complete plays which survive from ancient Greece. He is known to have written more than 90 plays, though only seven survive. The most famous of these are the trilogy known as Orestia. Also wellknown are The Persians and Prometheus Bound.	"The Persians" was written by Aeschylus.	Yes
SUM	A Pentagon committee and the congressionally chartered Iraq Study Group have been preparing reports for Bush, and Iran has asked the presidents of Iraq and Syria to meet in Tehran.	Bush will meet the presidents of Iraq and Syria in Tehran.	No

these resources only. Using only syntax enables to answer 34 per cent of the questions. The availability of a thesaurus raises this number to 48 per cent. By constrast, PETE solely focuses on syntax.

2.2. Dependency-Based Parser Evaluation

This section reviews some of the previous work in parser evaluation, with an emphasis on dependency-based methods.

A survey on parser evaluation is given by Carroll et al. [12]. They summarize the state-of-the-art up to 1998. Evaluation methods can be classified as non-corpus and corpus-based. Corpus-based methods are further divided based on whether the corpus is annotated or unannotated. Coverage methods measure the percentage of sentences in an unannotated corpus that receive at least one parse. A downside is that a parser returning trivial parses for every sentence would still score high in this method. Average Parse Base method calculates the geometric mean of the number of analyses divided by the number of tokens in a sentence. It shows the amount of ambiguity in the parser grammar, plotted against the sentence length. Its disadvantage is that a low coverage parser would perform well in this measure if the parses are relatively unambiguous. Entropy/Perplexity method applies a probabilistic language model on unannotated corpus and finds out how much a parser captures regularities and decreases ambiguity. Part-of-speech Assignment Accuracy has the advantage that there is already a large amount of part-of-speech tagged corpus. However, many parsers take pre-tagged corpus as input and applicability of this method is low.

They also propose grammatical relations (GR) for parser evaluation [12]. GRs are arranged in a LFG-like notation. A question mark is placed for unspecified information. All possible derivations out of a parse are computed. Evaluation is done by measuring precision and recall on the set of manually annotated GRs. Basically, a grammatical relation shows the syntantic dependency between a head and a dependent. GRs are organized hierarchically. For evaluation SUSANNE corpus was converted into GR scheme by first automatic processing and then manual inspection. On average there

are around four GRs per sentence.

Lin [13] converts phrase structures into dependency representations. The proposed evaluation is illustrated using MiniPar, which follows the Minimalist Program. Dependency relations are asymmetric binary relations between a head and a modifier. To represent the dependency tree, a tuple is written for each word. Precision and recall metrics are calculated by comparing tuples in gold standard and response sentences. Dependency labels are ignored.

Gaizauskas et al. [14] propose flatter annotation structures that would have relatively more consensus across grammatical theories. A two-step approach is proposed. First, problematic items are deleted. Second, transformational rules are used to convert annotations into canonical forms. Relations that are common to different parsers are considered. Due to omissions, sentence structures are flatter. Authors acknowledge that divergences may still exist, therefore an additional post-processing step on the parser output may be required. An advantage is that it is easier to develop a corpus with flatter structures. Recall and conformance metrics are used for evaluation. Within this framework the precision metric is unsuitable because the gold standard annotation is minimal. The conformance metric is defined as the proportion of gold standard constituents that are not crossed by any constituent in the response.

De Marneffe et al. [15] extract dependency relations from phrase structure parses. Stanford dependency (SD) relations are similar to GRs [12]. Tree regular expressions are used to produce typed dependencies. Dependencies are extracted and then dependency types are determined. Semantic heads are retrieved instead of syntactic heads. Furthermore, some words such as prepositions and conjunctions are collapsed. Additional links may be inserted for better semantic interpretability. They perform a small-scale evaluation between MiniPar, Link and Stanford parsers using 10 sentences from the Brown corpus. However, such an evaluation is difficult because their dependency representations are different.

Similarly, Miyao et al. [7] convert parser output to GR and SD representations.

Conversion is done as a post processing step. Conversion from the Penn Treebank is approximate and problematic. Heuristic approaches are used to overcome problems. Although SD and GR look similar, conversion between them is not trivial either. 560 sentences, annotated in both GR and SD schemes, from the Penn Treebank test section are used in the experiments. It is estimated that 20 per cent of conversions are problematic.

Clark and Curran [16] evaluate a CCG parser using GRs. Each of 425 lexical categories is mapped to a GR. A post-processing step handles the remaining discrepancies. Gold standard CCGBank data have been used for the development of transformation rules. Evaluation is done on 560 test sentences from the CCGBank version of Penn Treebank. The upper bound for a score that can be obtained in the evaluation is 84 per cent.

In order to get around conversion problems, Tam et al. [17] propose that linguistic features be collected from the parser output and compared to the gold standard. Annotators list the linguistic phenomena for the gold standard set. The most salient phenomenon is taken. Additionally, annotators write the most likely error that parsers would make for each sentence. A recognizer lists the linguistic phenomena in the parser output. Points are given if the recognizer list contains the correct phenomenon or if it does not contain the incorrect phenomenon. A small scale evaluation consisting of 10 sentences is carried out. Its disadvantage is that the linguistic phenomenon may not be recognized even if the parser output is correct.

Rimell and Clark [18] analyze the decisions made by several GR-like evaluation schemes. A decision involves constructions such as subject of verb, direct object of verb, passive voice. Another decision is choosing words that enter a particular construction. A related decision is choosing words to represent the construction. Usually a few words that are deemed important represent a construction that may span more words. Lastly a decision involves the choice of arguments. Empirical study is needed to show which decisions are better in different situations.

Miyao et al. [19] give a task-oriented parser evaluation. They compare recent parsers from different frameworks in a practical application. In total eight dependency, phrase-structure and deep parsers are evaluated. Another goal is to analyze the domain-adaptibility of these data-driven parsers. The task-oriented method uses parser outputs in a machine learning classifier to detect protein–protein interactions. Parse tree indicates a interaction if there is a close syntactic relationship between lexical items. All parsers achieved higher scores than the baseline of bag-of-words. Parser accuracies are similar. Accuracies increase after training with domain specific data. Parsing times vary significantly. Dependency parsing is the fastest, phrase-structure parsing is the slowest and deep parsing is in between. As for the efficacy of the representation, CoNLL format seems to be better than the Penn Treebank format. An ensemble of parsers results in higher accuracy values.

3. STATISTICAL PARSING

This chapter describes input formats, evaluation metrics, and algorithms in phrase-structure and dependency parsing. Parsers evaluated in this study are introduced.

3.1. Phrase-Structure Parsing

Phrase-structure grammars make use of the constituency concept. A constituent is a group of words behaving as a single unit [1]. An example is the noun phrase "the red apple".

3.1.1. Input Format

Data are represented in the Penn Treebank via Lisp-style parentheses. Internal node labes follow opening parentheses. Terminal nodes are enclosed within a pair of parentheses. A node with a smaller indentation level is located higher in the parse tree.

```
(S
  (NP-SBJ
    (NP (RB Not) (PDT all) (DT those) )
    (SBAR
      (WHNP-3 (WP who) )
      (S
        (NP-SBJ (-NONE- *T*-3) )
        (VP (VBD wrote) ))))
  (VP (VBP oppose)
    (NP (DT the) (NNS changes) ))))
```

Figure 3.1. Penn Treebank data

Figure 3.2 shows the visual representation of the parse tree given in Figure 3.1. This parse tree contains function tags such as NP-SBJ and empty nodes indicating a wh-trace. These fields are removed in the training data. Parsers do not include them in their output as well.

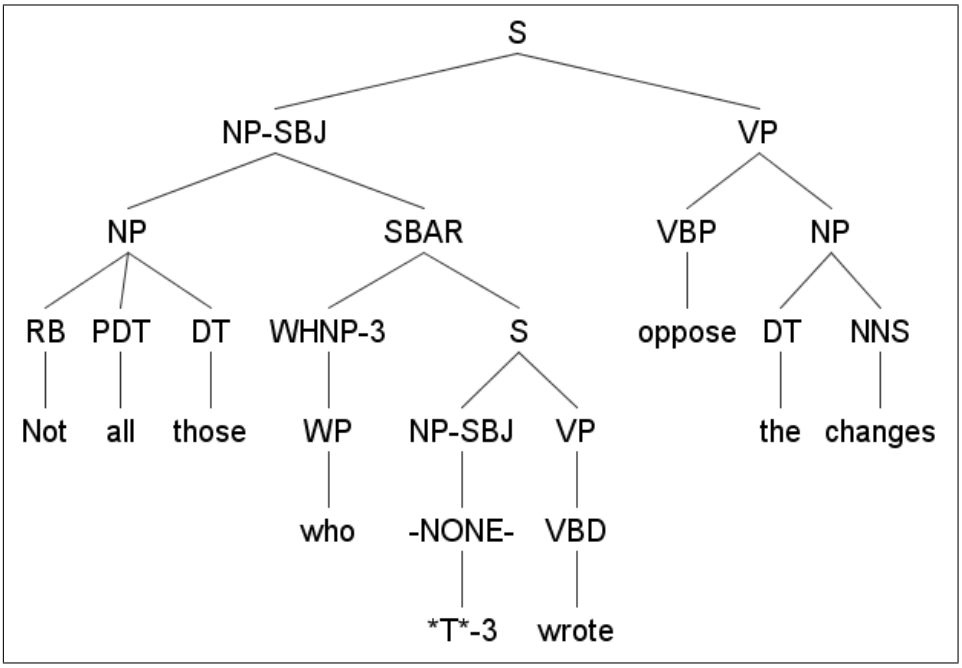

Figure 3.2. Penn Treebank parse tree

3.1.2. Evaluation Metrics

The standard evaluation method for the phrase-structure framework has been the bracketing precision and recall [20]. A constituent assignment in the parser output is deemed correct if the span of words and and the label are the same as in the gold standard parse. The crossing brackets score counts the number of parentheses that are only partially contained within a pair of parentheses in the gold standard parse.

$$\text{Recall} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in treebank parse}} \tag{3.1}$$

$$\text{Precision} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in proposed parse}} \tag{3.2}$$

F-measure is the harmonic mean of recall and precision.

$$\text{F-measure} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{3.3}$$

Crossing brackets metric is defined similarly [21].

$$\begin{aligned}\text{Crossing Brackets} = &\text{number of constituents which violate constituent} \\ &\text{boundaries with a constituent in the treebank parse}\end{aligned} \tag{3.4}$$

3.1.3. General Approach

A probabilistic context-free grammar (PCFG) has probabilities for the derivational rules, which are calculated by referring to the treebank. Given a sentence S and a parse T, a generative model derives the probability $P(T, S)$.

$$P(T, S) = \prod_{i=1}^{n} P(RHS_i | LHS_i) \tag{3.5}$$

Parsing a sentence amounts to finding the most probable parse T given a sentence S. To efficiently search through possible parses, dynamic programming is used. Most of the state-of-the-art parsers use Cocke-Kasami-Younger (CKY) or other chart parsing algorithms. Maximizing the conditional probability is equivalent to maximizing the joint probability or the probability of the parse as shown in the following equation. Basically parsers differ in how they formulate $P(T)$.

$$
\begin{aligned}
T_{best} &= \arg\max_{T} P(T|S) \\
&= \arg\max_{T} \frac{P(T,S)}{P(S)} \\
&= \arg\max_{T} P(T,S) \\
&= \arg\max_{T} P(T)
\end{aligned}
\tag{3.6}
$$

3.1.4. Charniak Parser

Charniak applies a maximum-entropy-inspired model to combine different conditioning events successfully [22]. Right-hand side labels are expanded with a third-order Markov grammar. The advantage of maximum-entropy models is that features are easily changeable and they need not be independent. The model is not exactly a maximum-entropy model since the partition function is not defined. Deleted interpolation is used to smooth the probabilities. A bottom-up best-first probabilistic chart parser finds the candidate parses. Guessing the pre-terminal before guessing the lexical head brings two per cent improvement.

3.1.5. Collins Parser

Collins proposes three generative lexicalized models [21]. Model 1 is a basic generative model. Right-hand side labels are expanded by first generating the head and then the left and right modifiers. Zeroth-order Markov assumption is made. Distance function considers words between the head word and the edge of the constituent. Independence assumption for complements often cause parse errors. Model 2 addresses this problem by attaching a suffix to node labels of complements. Subcategorization frames are included in the probability model. Model 3 additionally considers wh-movement to prevent its negative effect on subcategorization probabilities. Several levels of back-off are defined to smooth the probabilities. A CKY style chart parser searches for the most probable parse.

3.1.6. Stanford Parser

A distinction between lexicalized and unlexicalized grammars is made based on whether non-terminals are annotated with head words. Unlike Charniak and Collins parsers, Stanford parser is unlexicalized. While its accuracy is a little lower than the lexicalized parsers, it has some advantages. It is easier to interpret and develop. The grammar is more compact. Parsing time complexity is smaller. A generalized CKY parser is used. Grammar is markovized with a vertical and horizontal order of two. Non-terminals are annotated with external and internal features. Only function words are included in probability calculations.

3.2. Dependency Parsing

Dependency grammars take a different approach. Intermediate level concepts such as constituents are not used. Rather the focus is directly on relations between words. The syntactic analysis of sentence is represented by a dependency tree, which is a labeled directed acyclic graph [23]. Words are represented as nodes in the graph. Each word is connected to a single head. An arc is non-projective if a word between the dependent and the head is not dominated by the head.

Dependency formalism has gained importance in recent years [24]. Possible applications include relation extraction, paraphrase acquisition and machine translation.

3.2.1. Input Format

There are several alternatives to represent a dependency parse, such as an XML-based representation. In this thesis, the CoNLL format [25] is used since it is widely supported by existing software. Each word is described on a single line using 10 tab-separated fields. One empty line delimits two consecutive sentences in the data file. Data fields are given in Table 3.1. Empty fields are denoted by underscores. Typically fields 6, 9, and 10 are not used. Id field starts at 1 for each sentence. Head field gives the id value of the head. Space characters are not allowed within fields.

Table 3.1. CoNLL data format

Field #	**Name**	**Description**
1	ID	Token counter
2	FORM	Word form
3	LEMMA	Lemma of word form
4	CPOSTAG	Coarse-grained part-of-speech tag
5	POSTAG	Fine-grained part-of-speech tag
6	FEATS	Syntactic or morphological features
7	HEAD	Head of the token
8	DEPREL	Dependency relation to head
9	PHEAD	Projective head
10	PDEPREL	Dependency relation to phead

Figure 3.3 gives the dependency parse data. Main verb of the sentence is connected to the artificial root node which has the id value of 0. Figure 3.4 shows the dependency tree representation. Unlike the phrase-structure representation of the same sentence in Figure 3.2, there are no empty nodes. Subject function tags are used in the dependency relations.

```
1   Not       not      RB   RB    _   6   SBJ    _   _
2   all       all      PD   PDT   _   1   NMOD   _   _
3   those     that     DT   DT    _   1   NMOD   _   _
4   who       who      WP   WP    _   5   SBJ    _   _
5   wrote     write    VB   VBD   _   1   NMOD   _   _
6   oppose    oppose   VB   VBP   _   0   ROOT   _   _
7   the       the      DT   DT    _   8   NMOD   _   _
8   changes   change   NN   NNS   _   6   OBJ    _   _
```

Figure 3.3. Dependency data

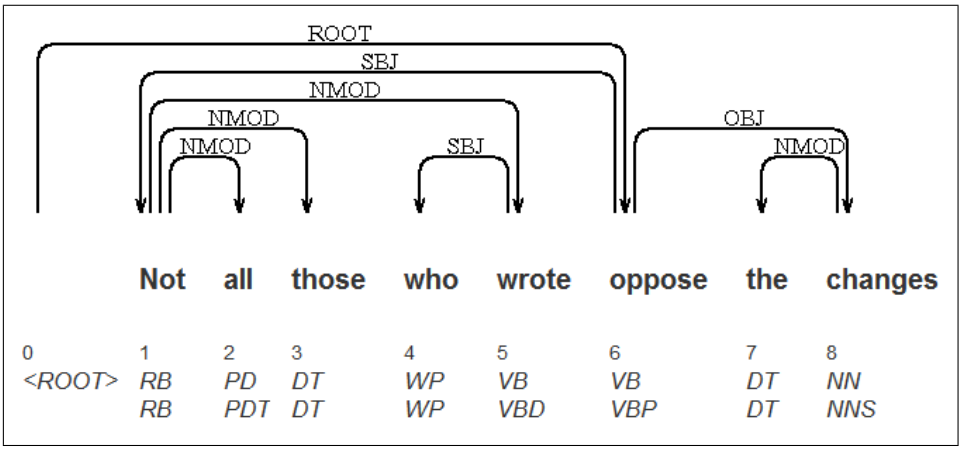

Figure 3.4. Dependency parse tree

3.2.2. Evaluation Metrics

The standard evaluation metrics for dependency parsing are Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS). UAS gives the percentage of words with the correct head. LAS gives the percentage of words with the correct head and the dependency label.

3.2.3. MSTParser

McDonald et al. use online large-margin learning to train a dependency parser [24]. It is based on a global approach since the accuracy of the overall tree is maximized. The score of a dependency tree is the sum of the scores of all edges in the tree. When parsing is cast as a multi-class classification, dependency trees correspond to classes for a sentence. Since there are exponentially many dependency trees, only k-best trees are considered. The disadvantage is that features are defined over single dependency attachments. A lexicalized CKY chart parser is used.

3.2.4. MaltParser

MaltParser is a transition-based dependency parser. In contrast to MSTParser, it carries out a local, greedy search. Partially processed tokens are stored in a stack

and unseen words in a list. A deterministic parsing algorithm decides whether a link should be created for the current word. Previous parser actions are considered in the history-based feature model. Discriminative machine learning maps histories to parser actions. MaltParser is desirable because it has linear parsing time complexity in projective mode. A problem is error propagation where later parsing decisions are adversely affected from early parsing errors.

4. GENERATING ENTAILMENTS

In PETE, entailments are designed as minimal sentences, each focusing on one relation. Entailments always include the head and the modifier, plus auxiliary words as necessary for grammaticality. This chapter describes how entailments are generated from dependency parses and phrase-structure parses.

4.1. Generating Entailments From Dependency Parses

Given a dependency parse, the program scans the sentence from left to right. For each word, a decision is made as to whether an entailment should be generated. Words such as determiners are skipped. Pseudocode of the entailment generation algorithm is listed in 4.1.

Input: D a dependency parse tree
Output: E a list of entailments
  for all word w in D **do**
    if w is not a skip word **then**
      $h \leftarrow$ head word of w
      if h is a verb **then**
        $e \leftarrow$ "subject(h) verb(h) object(h) adverb(h)"
      else if w is appositive **then**
        $e \leftarrow$ "h is w"
      else
        $e \leftarrow$ "there is w h"
      end if
      $E \leftarrow E \cup \{e\}$
    end if
  end for

Figure 4.1. Pseudocode for generating entailments from dependency parses

Generally function words are skipped. These include prepositions, the negation adverb not, conjunctions, particles, modals, and determiners. Verbs other than the last verb in a verb chain are skipped. Adjective modifiers of nouns are always included in the entailments generated for their head words. Therefore they are skipped.

An entailment is generated with the current word as the target word. A template is selected based on the target word or its head. If the head word is a verb, a simple sentence is formed including the head and the target words. If the target word is an appositive to the head word, a predicative sentence is formed. In the case where the target word is a nominal modifier other than an appositive, an existential construct is formed. Entailments are reorderings of the source sentence words. Few extra words, such as copulae, are added.

Examples for each entailment type are listed below. The first group is for the simple sentence type.

- But Mr. Lane said that while the SEC regulates who files, the law tells them when to do so.
  - X regulates who files.
- The SEC will probably vote on the proposal early next year, he said.
  - X will vote early next year.
- The proposed changes also would allow executives to report exercises of options later and less often.
  - X would allow executives to report X.

The second group of examples is for predicative entailments. Appositive is a fairly frequent construct in the Penn Treebank.

- Bell, based in Los Angeles, makes and distributes electronic, computer and building products.
  - Bell was based.
- The proposed rules also would be tougher on the insiders still required to file

reports, he said.

– The insiders are required to file X.

The last group of examples is for existential entailments. Target words are nominal modifiers.

- The rules will eliminate filings policy-making divisions, such as sales, marketing, finance and research and development, Mr. Lane said.
  – There are divisions such as research.
- The luxury auto maker last year sold 1,214 cars in the U.S.
  – There is the luxury auto maker.
- Bell Industries Inc. increased its quarterly to 10 cents from seven cents a share.
  – There are seven cents a share.
- A SEC proposal to ease reporting requirements for some company executives would undermine the usefulness of information on insider trades as a stock-picking tool, individual investors and professional money managers contend.
  – There is a proposal to ease X.

A point worth mentioning is relative clauses. Heuristic rules are used to replace the relative pronoun with the head noun. While these methods work correctly most of the time, there are some limitations [21].

4.2. Generating Entailments From Phrase-Structure Parses

There is not a separate program for generating entailments from phrase-structure parses, nor there is any reason to do so. Having a program mimic another program's output is not a trivial task. That would cause unnecessary differences in the generated entailments, and eventually lost points in the evaluation. Instead, phrase-structure parses are converted to dependency parses using the standard conversion program [26] and the dependency program of Section 4.1 is used to generate entailments.

Some information is lost in the conversion. Coordination ambiguity arises due to

Melcuk style analysis of coordination [27] in dependency parses. Modification scope can be determined in phrase-structure parses by looking at the attachment level. However, in dependency parses a dependent is always linked to the first conjunct, irrespective of whether it modifies only its immediate head or all conjuncts.

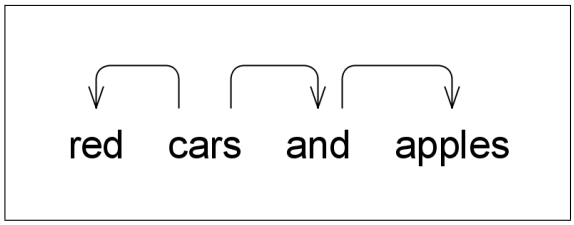

Figure 4.2. Coordination in dependency representation

Figure 4.2 illustrates the coordination ambiguity problem. According to this dependency parse, cars are red. However, it is unclear whether apples are red. Although there are alternative coordination styles that avoid coordination ambiguity, Melcuk style is preferred because links to a conjunction word are harder to learn for data-driven dependency parsers.

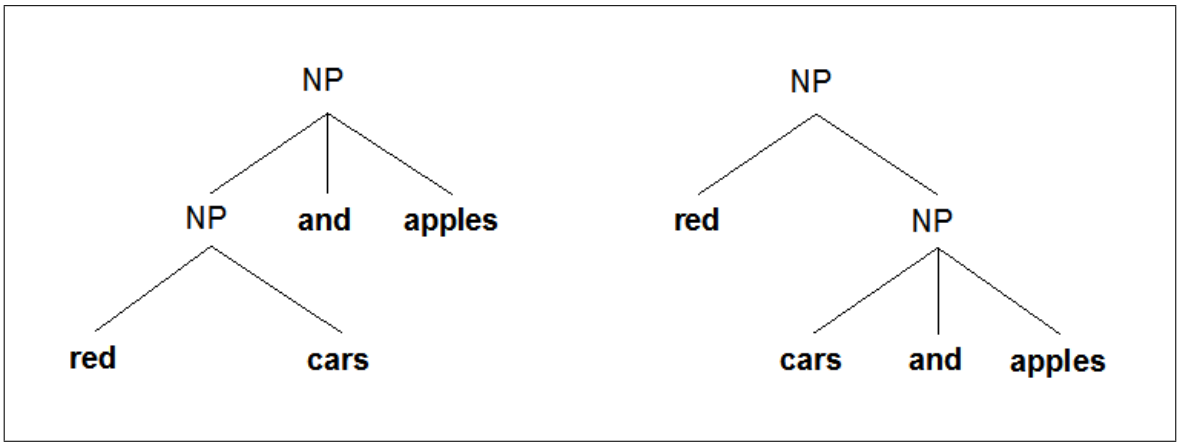

Figure 4.3. Coordination in phrase-structure representation

Figure 4.3 shows two simplified coordination examples in phrase structure representation with different attachment levels. In the left parse tree, red modifies only cars. On the right one, red modifies both cars and apples. After conversion, these two parses map to a single dependency parse, effectively resulting in an ambiguity.

Another point to consider for the conversion is the output format. The Penn Treebank [3] uses function tags such as SBJ (subject) and TMP (temporal) to give additional syntactic and semantic information. Empty nodes show non-local dependencies. Most phrase-structure parsers output a skeletal syntactic structure without function tags and empty nodes. Entailment generation program requires dependency relations, which must be output by the conversion program. The conversion program gets the dependency relations from the function tags of the parse input in turn. To generate entailments correctly either phrase-structure parses should be augmented with function tags or the dependency relations of the conversion output should be corrected.

Two heuristics are used to correct dependency relations. Subject heuristic tries to assign SBJ tag to a nominal dependent of a verb. Predicate heuristic tries to assign PRD tag to a nearest right dependent of a predicative verb. Both heuristics are a few lines of code. They are always applied for a fair evaluation of phrase-structure parsers.

Blaheta's function-tagger [28] is used. It is based on a relatively simple statistical method and is the first one to recover function tags. It takes skeletal parser output as input and appends function tags to some node labels. It is explicitly stated when the function-tagger is used in the experiments.

Figure 4.4 gives the algorithm that generates the additional entailments lost due to the coordination ambiguity after conversion. Subtrees of conjuncts are replaced to generate a new entailment. If a modifier modifies both conjuncts, it is at a higher level in the parse tree and it is not discarded with the first conjunct. Thus a new entailment showing the modification of the second conjunct is generated.

Input: D a dependency parse tree

P a phrase structure parse tree

E_D a list of entailments generated from D

Output: E a list of entailments

for all word w in D **do**

if $\exists$ entailment e for w in E_D **then**

$W \leftarrow$ words in e

$l \leftarrow$ lowest common ancestor of W in P

Check for coordination recursively starting at l

if $\exists$ node n having coordination **then**

$s_i \leftarrow$ subtree of the conjunct $c_i \in W$

$s_o \leftarrow$ subtree of the conjunct $c_o \notin W$

$W' \leftarrow$ remove s_i add s_o

$e' \leftarrow$ entailment with words W'

$E_P \leftarrow E_P \cup \{e'\}$

end if

end if

end for

$E \leftarrow E_D \cup E_P$

Figure 4.4. Generating entailments from phrase-structure parse trees

5. EVALUATION

State-of-the-art phrase-structure and dependency parsers are evaluated. Evalution metrics are defined in Chapter 3. Three phrase-structure parsers are Charniak parser [22], Bikel's implementation [29] of Collins parser [30], and Stanford parser [31]. Collins parser runs Collins Model 2. Stanford parser is unlexicalized, other two parsers are lexicalized. Pre-trained models with basic configurations are used.

Dependency parsers evaluated in this study are MSTParser [24] and MaltParser [23]. They were trained with the dependency data converted from Penn Treebank training section using the LTH converter [26]. Both parsers run in projective mode.

5.1. Evaluation on the WSJ Corpus

Table 5.1 shows the bracketing scores on the WSJ section 23. There are 2399 sentences in the test set. Sentence lengths are not considered in the reported results.

Table 5.1. Bracketing scores of phrase-structure parsers on the WSJ test section

System	**Recall**	**Precision**	**F-measure**
Charniak	89.57	89.91	89.74
Collins	88.13	88.26	88.19
Stanford	85.09	86.52	85.80

Table 5.2. LAS and UAS scores of dependency parsers on the WSJ test section

System	**UAS**	**LAS**
MST	92.0	88.7
Malt	89.8	86.8

Similarly, LAS and UAS scores are listed in Table 5.2.

Dependency-based evaluation of all parsers is given in Table 5.3. Phrase-structure parser outputs are converted to dependency representation using LTH converter. For dependency parsers, scores are the same as in Table 5.2. Charniak parser is best at getting the unlabeled dependency tree correct, as shown in the UAS column. The large gap between UAS and LAS scores show that phrase-structure parsers need to incorporate function tags in their outputs. The gap is 8.3 for Charniak parser and 3.3 for MSTParser. Blaheta's function-tagger does a good job by reducing the gap to less than 4.0. Table 5.3 is not a fair comparison since coordination information in phrase-structure parses is not considered.

Table 5.3. LAS and UAS scores on the WSJ test section

System	**UAS**	**LAS**	**UAS−LAS**
Charniak + F-Tags	93.2	89.6	3.6
Charniak	93.0	84.7	8.3
MST	92.0	88.7	3.3
Collins + F-Tags	91.6	87.7	3.9
Collins	91.4	83.1	8.3
Stanford + F-Tags	90.3	86.5	3.8
Stanford	90.2	82.1	8.1
Malt	89.8	86.8	3.0

PETE scores are given in Table 5.4. The gold standard set of entailments are generated by applying the algorithm in Figure 4.1 to the gold standard dependency parse and the algorithm in Figure 4.4 to the gold standard phrase-structure parse. The first three lines indicate upper bounds. Recall upper bound for dependency parsers is less than 100 per cent because they miss the additional coordination entailments generated from the gold standard phrase-structure parse. On the other hand, phrase-structure parsers lose points due to their impoverished output format. For second and third rows, function tags and empty nodes are removed from the gold standard phrase-structure parse. Inperfect conversion determines the upper bound in this case. The function-tagger increases the score by four per cent. In PETE evaluation, a full

phrase-structure parser [32], not included in this study, has 100 per cent upper bound.

Recall is more relevant than F-measure in this table. The decision of when to return an answer is generally systematically determined. Dependency parsers do not return an answer for the additional coordination entailments. Consequently their precision scores are higher than their recall scores. F-measure is relevant for parsers using the same framework.

Two phrase-structure parsers got the highest scores in PETE evaluation. The function-tagger improves the results, although its effect is less pronounced than in LAS scores. Charniak parser without the function-tagger performs better than the best performing dependency parser.

Table 5.4. PETE scores on the WSJ test section

System	**Recall**	**Precision**	**F-measure**
Gold Dep.	98.12	100.00	99.05
Gold Phrase + F-Tags	96.40	96.26	96.33
Gold Phrase	92.42	92.02	92.22
Charniak + F-Tags	83.34	83.35	83.34
Collins + F-Tags	81.60	82.16	81.88
Charniak	80.57	80.33	80.45
MST	79.24	81.17	80.19
Collins	78.99	79.35	79.17
Malt	77.66	80.82	79.21
Stanford + F-Tags	77.58	77.63	77.60
Stanford	75.10	75.02	75.06

5.2. Evaluation on the Brown Corpus

Whereas WSJ corpus of the Penn Treebank includes only financial texts, Brown corpus includes various genres such as fiction and natural sciences. The rationale for

evaluation on Brown corpus is to see how much parsers overfit to the WSJ data and how well they perform in a new domain. There are 425 sentences in the test set, the same sentences as in the CoNLL 2008 shared task data set [25].

Table 5.5 gives the bracketing scores. While the ranking is the same as in Table 5.1 the gap between Stanford parser and Collins parser has shrunk.

Table 5.5. Bracketing scores of phrase-structure parsers on the Brown test section

System	**Recall**	**Precision**	**F-measure**
Charniak	85.74	85.90	85.82
Collins	83.57	83.91	83.74
Stanford	83.00	83.01	83.00

Attachment scores of dependency parsers are given in Table 5.6. While the scores are lower than in Table 5.2, the ranking has not changed.

Table 5.6. LAS and UAS scores of dependency parsers on the Brown test section

System	**UAS**	**LAS**
MST	88.2	81.9
Malt	85.2	79.2

PETE scores are given in Table 5.7. Coordination ambiguity has a slightly larger role in the Brown corpus since the upper bound of recall for dependency parsing is 96.75 per cent compared to 98.12 per cent. There are some notable differences in the ranking. Charniak parser performs even better. Its score is still higher than other parsers without using the function-tagger. Stanford parser has risen in the ranking; probably its unlexicalized model helps avoid overfitting. Dependency parsers seem to be worst affected by the overfitting problem since the decline in scores cannot be explained by the decreased upper bound. Their rich feature models may make them more susceptible to overfitting.

Table 5.7. PETE scores on the Brown test section

System	**Recall**	**Precision**	**F-measure**
Gold Dep.	96.75	100.00	98.35
Gold Phrase + F-Tags	94.79	94.89	94.84
Gold Phrase	89.82	89.26	89.54
Charniak + F-Tags	74.89	74.64	74.76
Charniak	72.24	71.60	71.92
Collins + F-Tags	71.38	72.68	72.02
Collins	68.76	69.68	69.22
Stanford + F-Tags	68.23	68.61	68.42
MST	67.66	70.98	69.28
Stanford	66.60	66.78	66.69
Malt	65.87	71.44	68.54

6. CONCLUSION

PETE is proposed as a cross-framework parser evaluation scheme. State-of-the-art phrase-structure and dependency parsers are evaluated using PETE. A program generates entailments from dependency parses. Phrase-structure parses are converted to dependency parses to generate entailments. A goal is to provide a fair comparison. To this end, additional entailments are generated for unambiguous coordinations in phrase-structure parses. The ranking shows that dependency trees converted from lexicalized phrase-structure parser outputs are more accurate than native dependency parser outputs, in agreement with previous results [24].

Unlike other dependency-based evaluations, PETE is designed with annotation in mind. Traditionally, annotators go through a training period to acquire the intricacies of an annotation scheme. Limited number of annotators are usually based in the same institution. Consequently, annotation is time consuming and expensive. PETE offers a new annotation model. A priori training is not required. Many people can participate in the annotation process over the Web. In one recent work, a large number of non-expert annotators evaluated machine translation quality via Amazon's Mechanical Turk [33]. Although PETE annotation should not take significantly more time than the comprehension time of a simple sentence, trial annotations are required to compare the speed of annotation.

Another possible application of PETE is to "textualize" parser outputs. Benefits of visualization software are well known. In the parsing domain, one such software is MaltEval [34], which was used during this thesis to a great extent. While the general wisdom states that a picture is worth a thousand words, parse trees of sentences longer than a few dozen words can be daunting. To check parsing errors around a target word, one follows incoming and outcoming dependency links. This involves a scan for each argument over the dependency tree, which is inefficient. To understand a word's function in a sentence, one has to collect at least the head and any required arguments. PETE provides this information in the entailment generated for the target word. To

detect parsing errors, one checks whether the entailment has a correct grammatical structure and it makes sense according to the given sentence.

Evaluation methods are expected to provide a clear assessment of current technologies and directions for future research. The detrimental effect of impoverished output format of phrase-structure parsers have been noted. However, the standard bracketing evaluation does not create incentives for a richer output. Therefore most state-of-the-art phrase-structure parsers output only a skeletal parse tree. PETE clearly shows that incorporating functions tags and empty nodes is beneficial.

The natural language processing field has seen great advances since 1990's following the success of statistical systems. However, it is not clear how well those gains will translate to real world applications [35]. There is a visible shift toward semantically-relevant evaluations in recent years. One approach is task-oriented evaluation [19]. PETE does not commit to a linguistic representation other than the surface form. While the parser specifics are abstracted away, the actual parser decisions are more traceable than in a task-oriented evaluation.

APPENDIX A: SAMPLE ENTAILMENTS

- Ms. Haag plays Elianti.
  - There is Ms. Haag.
  - Ms. Haag plays X.
  - X plays Elianti.
- Rolls-Royce Motor Cars Inc. said it expects its U.S. sales to remain steady at about 1,200 cars in 1990.
  - There are Rolls-Royce Motor Cars Inc.
  - Rolls-Royce Motor Cars Inc. said X.
  - It expects X.
  - X said X expects X.
  - There are its U.S. sales.
  - X expects its sales to remain steady.
  - X expects X to remain steady.
  - X are to remain steady.
  - X are to remain X at about 1,200 cars.
  - X are to remain X in 1990.
- Companies would be compelled to publish in annual proxy statements the names of insiders who fail to file reports on time.
  - Companies would be compelled to publish X.
  - There are annual proxy statements.
  - X would be compelled to publish X in statements.
  - X would be compelled to publish the names.
  - There are the names of insiders.
  - Insiders fail to file X.
  - X fail to file reports.
  - X fail to file X on time.

REFERENCES

1. Jurafsky, D. and J. Martin, *Speech and Language Processing*, Prentice Hall, 2008.

2. Marcus, M., M. Marcinkiewicz, and B. Santorini, "Building a Large Annotated Corpus of English: the Penn Treebank", *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330, 1993.

3. Bies, A., M. Ferguson, K. Katz, and R. MacIntyre, "Bracketing Guidelines for Treebank II Style Penn Treebank Project", Technical report, University of Pennsylvania, 1995.

4. Marcus, M., G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger, "The Penn Treebank: Annotating Predicate Argument Structure", *ARPA Human Language Technology Workshop*, 1994.

5. Dickinson, M. and W. D. Meurers, "Detecting Inconsistencies in Treebanks", *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, Växjö, Sweden, 2003.

6. Yuret, D., "Why You Should Not Use the Penn Treebank to Train a Parser", http://denizyuret.blogspot.com/2006/10/why-you-should-not-use-penn-treebank.html, 2006.

7. Miyao, Y., K. Sagae, and J. Tsujii, "Towards Framework-Independent Evaluation of Deep Linguistic Parsers", King, T. H. and E. M. Bender (editors), *Proceedings of the Grammar Engineering Across Frameworks Workshop (GEAF-07)*, pp. 238–258, CSLI, Stanford, CA, 2007.

8. Dagan, I., O. Glickman, and B. Magnini, "The PASCAL Recognising Textual Entailment Challenge", Quinonero-Candela, J., I. Dagan, B. Magnini, and F. d'Alché Buc (editors), *Machine Learning Challenges. Lecture Notes in Computer Science*,

Vol. 3944, pp. 177–190, Springer, 2006.

9. Bar-Haim, R., I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, "The Second PASCAL Recognising Textual Entailment Challenge", *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pp. 1–9, 2006.

10. Giampiccolo, D., B. Magnini, I. Dagan, and B. Dolan, "The Third PASCAL Recognizing Textual Entailment Challenge", *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, 2007.

11. Vanderwende, L. and W. Dolan, "What Syntax Can Contribute in the Entailment Task", *Lecture Notes In Computer Science*, Vol. 3944, p. 205, 2006.

12. Carroll, J., T. Briscoe, and A. Sanfilippo, "Parser Evaluation: a Survey and a New Proposal", *Proceedings of LREC*, pp. 447–454, 1998.

13. Lin, D., "Dependency-Based Evaluation of MINIPAR", *Proceedings of the LREC Workshop on Evaluation of Parsing Systems*, Granada, Spain, 1998.

14. Gaizauskas, R., M. Hepple, and C. Huyck, "A Scheme for Comparative Evaluation of Diverse Parsing Systems", *Proceedings of LREC*, pp. 143–149, 1998.

15. de Marneffe, M., B. MacCartney, and C. Manning, "Generating Typed Dependency Parses From Phrase Structure Parses", *Proceedings of LREC*, 2006.

16. Clark, S. and J. Curran, "Formalism-Independent Parser Evaluation with CCG and DepBank", *Proceedings of ACL*, pp. 248–255, Association for Computational Linguistics, Prague, Czech Republic, June 2007.

17. Tam, W. L., Y. Sato, Y. Miyao, and J. Tsujii, "Parser Evaluation Across Frameworks without Format Conversion", *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pp. 29–35, Coling 2008 Organizing Committee, Manchester, UK, August 2008.

18. Rimell, L. and S. Clark, "Constructing a Parser Evaluation Scheme", *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pp. 44–50, Manchester, UK, August 2008.

19. Miyao, Y., R. Sætre, K. Sagae, T. Matsuzaki, and J. Tsujii, "Task-Oriented Evaluation of Syntactic Parsers and Their Representations", *Proceedings of ACL-HLT*, pp. 46–54, 2008.

20. Abney, S., S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, and M. Liberman, "Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", *Proceedings of the Workshop on Speech and Natural Language*, pp. 306–311, Association for Computational Linguistics Morristown, NJ, USA, 1991.

21. Collins, M., "Three Generative, Lexicalised Models for Statistical Parsing", *ACL*, pp. 16–23, Association for Computational Linguistics, Madrid, Spain, July 1997.

22. Charniak, E., "A Maximum-Entropy-Inspired Parser", *Proceedings of NAACL*, pp. 132–139, 2000.

23. Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi, "MaltParser: A language-independent system for data-driven dependency parsing", *Natural Language Engineering*, Vol. 13, No. 2, pp. 95–135, 2007.

24. McDonald, R., K. Crammer, and F. Pereira, "Online Large-Margin Training of Dependency Parsers", *Proceedings of ACL*, pp. 91–98, Association for Computational Linguistics, Ann Arbor, Michigan, June 2005.

25. Surdeanu, M., R. Johansson, A. Meyers, L. Màrquez, and J. Nivre, "The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies", *Proceedings of CoNLL*, 2008.

26. Johansson, R. and P. Nugues, "Extended Constituent-to-dependency Conversion

for English", *Proceedings of NODALIDA 2007*, Tartu, Estonia, May 25-26 2007.

27. Melcuk, I. A., *Dependency Syntax: Theory and Practice*, State University Press of New York, 1988.

28. Blaheta, D., *Function Tagging*, Ph.D. thesis, Brown University, 2003.

29. Bikel, D., "Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine", *Proceedings of HLT*, pp. 178–182, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2002.

30. Collins, M., "Head-Driven Statistical Models for Natural Language Parsing", *Computational Linguistics*, Vol. 29, No. 4, pp. 589–637, 2003.

31. Klein, D. and C. Manning, "Accurate Unlexicalized Parsing", *Proceedings of ACL*, pp. 423–430, Association for Computational Linguistics, 2003.

32. Gabbard, R., S. Kulick, and M. Marcus, "Fully Parsing The Penn Treebank", *Proceedings of HLT-NAACL*, pp. 184–191, 2006.

33. Callison-Burch, C., "Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk", *Proceedings of EMNLP*, pp. 286–295, Association for Computational Linguistics, Singapore, August 2009.

34. Nilsson, J. and J. Nivre, "MaltEval: An Evaluation and Visualization Tool for Dependency Parsing", *Proceedings of LREC*, 2008.

35. Belz, A., "That's Nice... What Can You Do With It?", *Computational Linguistics*, Vol. 35, No. 1, pp. 111–118, March 2009.