**International Journal of Document Analysis and Recognition (IJDAR) manuscript No.**
(will be inserted by the editor)

**Jennifer Foster**

# Treebanks Gone Bad

## Parser Evaluation and Retraining using a Treebank of Ungrammatical Sentences

Received: date / Accepted: date

**Abstract** This article describes how a treebank of ungrammatical sentences can be created from a treebank of well-formed sentences. The treebank creation procedure involves the automatic introduction of frequently occurring grammatical errors into the sentences in an existing treebank, and the minimal transformation of the original analyses in the treebank so that they describe the newly created ill-formed sentences. Such a treebank can be used to test how well a parser is able to ignore grammatical errors in texts (as people do), and can be used to induce a grammar capable of analysing such sentences. This article demonstrates these two applications using the Penn Treebank. In a robustness evaluation experiment, two state-of-the-art statistical parsers are evaluated on an ungrammatical version of Section 23 of the Wall Street Journal (WSJ) portion of the Penn Treebank. This experiment shows that the performance of both parsers degrades with grammatical noise. A breakdown by error type is provided for both parsers. A second experiment retrains both parsers using an ungrammatical version of WSJ Sections 2-21. This experiment indicates that an ungrammatical treebank is a useful resource in improving parser robustness to grammatical errors, but that the correct combination of grammatical and ungrammatical training data has yet to be determined.

**Keywords** Treebanks · Parser Evaluation · Robust Parsing · Ungrammatical Language

## 1 Introduction

If a parser is to play a useful role in a natural language processing application, it must be robust to noise in the form of grammatical errors. This robustness manifests itself at four increasingly informative levels:

1. **Some Analysis**:
   The parser returns an analysis (possibly a partial one) for the ungrammatical sentence.
2. **Correct Analysis**:
   The parser returns a full analysis for the ungrammatical sentence and this analysis reflects the intended meaning of the ungrammatical sentence.
3. **Correct Analysis + Grammaticality Judgement**:
   The parser returns a full analysis which reflects the intended meaning of the ungrammatical sentence, and it also recognises that an error has occurred somewhere within the sentence.
4. **Correct Analysis + Grammaticality Judgement + Error Correction**:
   The parser returns a full analysis which reflects the intended meaning of the ungrammatical sentence, it recognises that an error has occurred *and* it suggests a correction for the error.

If the parser is to be employed in a grammar checking or Computer-Assisted Language Learning system, the third level of robustness is desirable, and the fourth level even more so. The input to such systems is the users' own language, and they expect feedback on whether or not it is well-formed. If, on the other hand, the parser is being used to provide an analysis of a sentence that serves as a step on the way to capturing the sentence's intended meaning, for example, in a machine translation or question answering system, the third and fourth levels of robustness are not a prerequisite. However, the first level of robustness is not guaranteed to be useful for such systems. It is, of course, better than a brittle response of no parse at all, but the second level of robustness is superior, because at this level, an analysis is returned which captures the ungrammatical sentence's intended meaning. Assuming that the parser in question is reasonably accurate when faced with well-formed language, the second level of robustness is equivalent to Menzel's definition of robustness [33] as a system's reluctance to change its

J. Foster
National Centre for Language Technology
School of Computing
Dublin City University, Ireland
Tel.: +353-1-7005263
Fax: +353-1-7005442
E-mail: jfoster@computing.dcu.ie

output when the input becomes increasingly ill-formed. Intuitively, this second level of robustness is close to the way people typically react to common grammatical errors — they attend to the sentence's meaning as if the error did not exist. This article is concerned with the second level of robustness, in particular, with examining to what extent this level of robustness can be achieved within a treebank-trained statistical parsing paradigm, using a treebank of ungrammatical sentences.

Traditional symbolic approaches to parsing employ grammars which aim to describe well-formed sentences and explicitly reject ill-formed ones. In order to analyse extra-grammatical input (including ungrammatical input) various robust parsing techniques have been proposed: *constraint relaxation* in which the parser's grammar is made more lenient [15,42,21], *error anticipation* in which special mal-rules are employed to explicitly describe ungrammatical structures [38,2], *minimum-edit-distance* approaches in which the ungrammatical sentence is transformed until it can be parsed [43,29] and *parse-fitting* approaches in which partial parses are pieced together using heuristics [25,36]. The treebank-trained statistical parsers [8,12,5,9] of the last fifteen years are inherently robust at the first level of robustness, since they will return an analysis for almost any sequence of words. The robustness of these parsers comes from their ability to overgenerate. Unlike traditional parsers, treebank-trained statistical parsers are generally agnostic to the concept of grammaticality but since they are usually trained on high-quality texts such as the Wall Street Journal, it is not clear that they are able to provide an accurate analysis for an ungrammatical sentence and thus achieve the second level of robustness. This article attempts to explore the extent to which a treebank-trained statistical parser can ignore grammatical noise using the idea of an "ungrammatical treebank". The exploration takes the following form: an ungrammatical version of the Wall Street Journal (WSJ) section of the Penn Treebank [32,31] is created, this ungrammatical version of the WSJ corpus is divided in the usual way into test and training data, and two WSJ-trained parsers, Bikel's implementation of Collin's Model 2 parser [12, 5] and Charniak and Johnson's reranking parser [9] are then evaluated against the test section of the ungrammatical WSJ to investigate these parsers' resistance to grammatical noise. The usual English training set for the parsers is then replaced/augmented with parses from the ungrammatical version of the WSJ, the parsers are retrained and then tested again to investigate whether the parsers' resistance to grammatical noise can be improved without affecting their performance on well-formed sentences.

The article is organised as follows: the idea of an ungrammatical treebank is described in more detail in Section 2. Section 3 contains a description of the data of interest, i.e. ungrammatical language. An attempt is made to give a definition of an ungrammatical sentence, and examples of sentences which fall under this definition and those which do not are provided. The process of creating an ungrammatical treebank is described in Section 4. In Section 5, the parser evaluation and retraining experiments involving the ungrammatical version of the WSJ treebank are presented and discussed. Finally, Section 6 summarises the main points in this article and proposes potentially worthwhile further work in this area.

## 2 An Ungrammatical Treebank: Motivation and Background

A corpus of ungrammatical sentences is a useful resource, both as a source of evidence for the kind of ill-formed structures that tend to occur in language, and as a source of test and training data for parsers which aim to accurately analyse sentences containing grammatical errors. Since people are able to comprehend text containing grammatical errors, it is reasonable to expect a parser to behave in the same way. A corpus of ungrammatical sentences can take the form of a learner corpus [22,16], i.e. a corpus of sentences produced by non-native learners of the language, or a more general form of error corpus, created by scanning texts for errors [1,18]. Learner corpora are particularly useful in the study of second language acquisition since they provide insight into the difficulties faced by native speakers of a particular language when attempting to learn the corpus language. The more general form of error corpus is unconcerned with whether an error reflects linguistic competence or performance, it merely records that an error has occurred. Unfortunately, the compilation of both kinds of error corpus is a slow process, because it is not enough to merely collect a body of sentences, the grammaticality of each sentence must also be judged in order to determine whether an error has occurred. If an error has occurred, it then must be classified according to some error taxonomy.

A usefully large error corpus, in which every sentence is guaranteed to contain a grammatical error, can be quickly created by automatically introducing errors into a corpus of grammatical sentences. In order to ensure that this transformation process is rooted in linguistic reality, it should, of course, be based on an analysis of naturally produced grammatical errors. An interesting aspect of the automatically induced error corpus is its parallel nature, since the meaning of the ungrammatical sentence can be found by looking at its grammatical counterpart.

An even more useful resource for the devising and testing of robust parsers, is a treebank of ungrammatical sentences. In the same way that a treebank of grammatical sentences can be used as a source of test data to evaluate parser output, and as a source of training data to build a probability model, a treebank of ungrammatical sentences can be used to evaluate and possibly improve upon a parser's performance on ungrammatical

language. In the absence of such a treebank, previous work by the author [17] attempts to evaluate parser output on ungrammatical data by evaluating it against its own output on a corrected, grammatical version of the same data. Thus, the parser provides its own gold standard. Bigert et al. [4] adopt a similar approach by introducing artificial spelling errors into error-free text and then evaluating parsers and a part-of-speech tagger on this text using their performance on the error-free text as a reference. Similarly, Lopresti [30] evaluates the effect of OCR errors on various NLP tasks such as sentence boundary detection and part-of-speech tagging by calculating the minimum-edit-distance between the original document and the noisy OCRed document. A drawback of these approaches to robustness evaluation is that an application's performance on noisy data is always evaluated against its performance on a well-formed version of the same data. It can happen that the application (parser, tagger, etc.) can produce an accurate analysis for the ill-formed input but not for the equivalent well-formed input (an example is provided by Foster [17]) and the evaluation metric must incorporate some kind of estimate of how often this is likely to occur. An ungrammatical treebank bypasses this problem because it serves as a stand-alone and accurate set of reference parses for ungrammatical sentences.

Of course, the creation of a treebank is a costly, laborious task. However, assuming the existence of a treebank of grammatical sentences and a corpus of ungrammatical sentences derived automatically from the sentences in the grammatical treebank, it is possible to automatically create a treebank of ungrammatical sentences. This treebank can then be partitioned in the usual way, into a set of gold standard reference parses and a set of training parses for any data-driven probabilistic parser.

The idea of an automatically generated error corpus is not new. Bigert et al [3,4], for example, automatically introduce spelling errors into texts. Okanohara and Tsujii [35] generate ill-formed sentences (they use the term "pseudo-negative examples") using a n-gram language model and they then train a discriminative language model to tell the difference between these pseudo-negative examples and well-formed sentences. Smith and Eisner [39,40] automatically generate ill-formed sentences by transposing or removing words within well-formed sentences. These ill-formed sentences are employed in a unsupervised learning technique called contrastive estimation which is used for part-of-speech tagging and dependency grammar induction. The idea of a treebank of ungrammatical sentences has been explored before by Kepser et al [28], who are responsible for compiling SINBAD, a treebank of German sentences which have been judged to be grammatically deviant by linguists. The SINBAD treebank differs from the type of ungrammatical treebank which would be produced by the method described here because it is designed to be used more as an informational source for generative linguists rather than as a set of training/test data for a robust parser. It is created manually rather than automatically, and is, thus, limited in size.

## 3 The Data of Interest: Ungrammatical Language

It is difficult to provide a satisfactory definition of the term "ungrammatical": for the purposes of this research, a sentence is defined to be ungrammatical if all the words in the sentence are well-formed words of the language in question, but the sentence contains one or more error [18]. Although this definition simply defines ungrammaticality in terms of error, it is less circular than one in which an ungrammatical sentence is defined to be a sentence which cannot be generated by the grammar of the language. An error can take the form of a performance slip which can occur due to carelessness or tiredness, or a competence error which occurs due to a lack of knowledge of a particular construction. This definition includes real-word or context-sensitive spelling errors and excludes non-word spelling errors. It also excludes the abbreviated informal language used in electronic communication [14,10,13]. For example, given the well-formed sentence (1) and the above definition of ungrammatical, sentences (2) and (3) are ungrammatical, whereas sentences (4) and (5) are not. Sentence (2) is ungrammatical (according to the definition) because it contains a real-word spelling error and sentence (3) is ungrammatical because it violates a well-defined word order constraint of English. Sentence (4) contains an error (a non-word spelling error) but since not all the words in the sentence are well-formed words of the English language, it is not ungrammatical according to the above definition. Finally, sentence (5) is not ungrammatical because it does not contain an error: the omission of the first person subject pronoun and the abbreviation of *be* to *b* are well-formed according to the norms of SMS communication.

(1) *I will be in town soon*
(2) *I will be* ***it*** *town soon*
(3) *I will in town soon* ***be***
(4) *I will be in town* ***soonn***
(5) ***Will b*** *in town soon*

Previous work by the author [19,20,18] involved the collection of ungrammatical written sentences in the English of newspapers, academic papers, emails and website forums. The resulting 20,000 word corpus was analysed and the following frequency ordering of the three word-level correction operators used to correct a grammatical error was found:

*substitute* (48%) > *insert* (24%) > *delete* (17%) > *combination* (11%)

The same ordering of the substitution, deletion and insertion correction operators was found in a study of native speaker spoken language slips carried out by Stemberger [41]. Foster [18] found that among the grammatical errors which could be corrected by substituting one word for another (48% of total), the most common errors were real-word spelling errors such as (2) above (20%), agreement errors (9%) and errors in verb form (5%). In fact, 75% of all errors fall into one of the following five classes:

(6) missing word errors:
*She didn't **want** to face him → She didn't to face him*
(7) extra word errors:
*Do you ever go and visit **any** of them? → Do you ever go and visit **the any** of them?*
(8) real-word spelling errors:
*I love **them** both → I love **then** both*
(9) agreement errors:
*The **contrast was** startling → The **contrasts was** startling*
(10) verb form errors:
*Want to **save** money? → Want to **saving** money?*

A similar classification was adopted by Nicholls [34], having carried out an error analysis on a learner corpus. Different languages and text types will exhibit a different error density and distribution. Hashemi [23], for example, finds that verb form errors are more common than agreement errors in a study of the written language of native Swedish speaking children, whereas agreement errors are more common than verb form errors in the English corpus upon which this research is based. Although not a truly representative sample, the corpus described by Foster is compiled from a sufficiently broad array of text types for us to conclude that the most common errors occurring within it are likely to occur elsewhere and for it to form the basis of the artificial error creation procedure described in the next section.

## 4 Creating an Ungrammatical Treebank

This section describes the procedure for creating an ungrammatical treebank. This procedure involves two steps: the first is the introduction of grammatical errors into the sentences in a treebank; the second is the transformation of the original gold standard analyses into gold standard analyses for the newly created ungrammatical sentences. The first step is described in Section 4.1, and the second in Section 4.2.

### 4.1 Automatic Error Creation

The error creation procedure takes as input a part-of-speech tagged corpus of sentences which are assumed to be well-formed, and outputs a part-of-speech tagged corpus of ungrammatical sentences. The error creation procedure is inspired by the manually created error corpus created by Foster [19,20,18], and the automatically introduced errors take the form of the five most common error types found in this corpus and introduced in Section 3, i.e. missing word errors, extra word errors, real-word spelling errors, agreement errors and verb form errors. The error creation procedure can be applied to its own output to yield sentences with more complex errors or with more than one of the above errors.

```
for each POS-tagged sentence, s, in input corpus do
  s_ungram ← ϵ
  missing_tried ← false
  extra_tried ← false
  real_word_spell_tried ← false
  agree_tried ← false
  verb_form_tried ← false
  while s_ungram is empty do
    n ← random float in range 0...1
    if n < missing_freq and missing_tried = false
    then
      s_ungram ← insert_missing_word_error(s)
      missing_tried ← true
    else if n < (missing_freq + extra_freq) and
    extra_tried = false then
      s_ungram ← insert_extra_word_error(s)
      extra_tried ← true
    else if n < (missing_freq + extra_freq +
    real_word_spell_freq) and real_word_spell_tried =
    false then
      s_ungram ← insert_real_word_spell_error(s)
      real_word_spell_tried ← true
    else if n ≤ (missing_freq + extra_freq +
    real_word_spell_freq+agree_freq) and agree_tried =
    false then
      s_ungram ← insert_agree_error(s)
      agree_tried ← true
    else if verb_form_tried = false then
      s_ungram ← insert_verb_form_error(s)
      verb_form_tried ← true
```

**Fig. 1** Top-Level Error Creation Algorithm

The top-level algorithm for creating the error corpus is shown in Fig. 1. The distribution of the five error types is established by setting the values of the four variables *missing_freq*, *extra_freq*, *real_word_spell_freq* and *agree_freq*. For this research, these relative frequency values are set to approximate the distribution found in the manually created error corpus (see Section 3).

#### *4.1.1 Missing Word Errors*

Missing word errors can be classified on the basis of the part of speech of the missing word. In the error corpus described by Foster [18], 98% of the missing word errors involve the omission of the following parts of speech (ordered by decreasing frequency)[1]

[1] Because the sentences in the manually created error corpus were encountered in context, it was sometimes possible to detect the erroneous omission of an adjective or an adverb. These are not included in the procedure for creating missing

**insert_missing_word_error(s)**
$s\_ungram \leftarrow \epsilon$
**if** $s$ contains more than one word **then**
$missing\_pos \leftarrow choose\_missing\_pos(s)$
**if** $missing\_pos$ is not empty **then**
$candidate\_list \leftarrow$ list of sentence positions, $p$, such that $\forall p \in candidate\_list$, POS(word@$p$ in $s$) = $missing\_pos$
$p' \leftarrow$ randomly selected position from $candidate\_list$
$s\_ungram \leftarrow s$ with $word@p'$ removed
return $s\_ungram$

**choose_missing_pos(s)**
$l \leftarrow < det, verb, prep, pronoun, noun, inf\_marker, conj >$
$l' \leftarrow l \cap$ set of all POS tags in $s$
$total\_freq \leftarrow \sum_{i=0}^{lengthofl'} freq(l'[i])$
**if** $l' \neq <>$ **then**
$next \leftarrow 0$
$n \leftarrow$ random float in range 0...1
**for** all POS tags, $pos \in l'$ **do**
$next \leftarrow next + freq[pos]/total\_freq$
**if** $n < next$ **then**
return $pos$
**else**
return $\epsilon$

**Fig. 2** Missing Word Error Creation Algorithm

*det(28%) > verb(23%) > prep(21%) > pro(10%) > noun(7%) > "to"(7%) > conj(2%)*

Missing word errors are introduced by searching a part-of-speech tagged sentence for all occurrences of words with the above part-of-speech tags and then deleting one from the sentence. The frequency ordering shown above is respected so that the resulting error corpus will contain, for example, more missing determiners than missing pronouns. In the unlikely event that a sentence contains none of the above parts of speech, no ungrammatical sentence is produced. Another case where no ungrammatical sentence is produced occurs when the input to the procedure is a one-word sentence such as *Yes*.

The algorithm for creating missing word errors is detailed in Fig. 2. The algorithm assumes the available of missing part-of-speech tag frequencies. As with the top-level algorithm, these are set according to the distribution found in the manually created corpus.

### *4.1.2 Extra Word Errors*

Extra word errors are introduced in the following three ways:

1. Random duplication of a token within a sentence: *That's the way **we we** learn here.*
2. Random duplication of a POS within a sentence: *There **it he** was.*
3. Random insertion of an arbitrary token into the sentence: *Joanna drew **as** a long breadth.*

word errors because their omission will systematically result in a grammatical sentence.

**insert_extra_word_error(s)**
$s\_ungram \leftarrow \epsilon$
$list \leftarrow$ list of tagged words from BNC subset
$n \leftarrow$ random integer in range 0..2
**if** $n = 0$ **then**
$s\_ungram \leftarrow introduce\_repeated\_token(s)$
**else if** $n = 1$ **then**
$s\_ungram \leftarrow introduce\_repeated\_tag(s, list)$
**else**
$s\_ungram \leftarrow introduce\_unnecessary\_word(s, list)$
**if** $s\_ungram$ is empty **then**
$s\_ungram \leftarrow introduce\_unnecessary\_word(s, list)$
return $s\_ungram$

**introduce_repeated_token(s)**
$extra \leftarrow \epsilon$
**while** $extra$ is empty and not all words tried **do**
$p \leftarrow$ random integer in range 1...$length(s)$
**if** POS(word@$p$ in $s$)$\neq$ $adj$ **then**
$extra \leftarrow$ word@$p$ in $s$
return $s$ with $extra$ inserted at position $p + 1$

**introduce_repeated_tag(s,list)**
$extra \leftarrow \epsilon$
**while** $extra$ is empty and not all words tried **do**
$p \leftarrow$ random integer in range 1...$length(s)$
**if** POS(word@$p$ in $s$)$\neq$ $adj$ **then**
$extra \leftarrow$ randomly selected word in $list$ where POS($extra$) = POS(word@$p$ in $s$)
**if** $extra$ is not empty **then**
return $s$ with $extra$ inserted at position $p + 1$
return $\epsilon$

**introduce_unnecessary_word(s,list)**
$p \leftarrow$ randomly selected integer in range 1...$length(s)$
$extra \leftarrow$ randomly selected word in $list$
return $s$ with $extra$ inserted at position $p + 1$

**Fig. 3** Extra Word Error Creation Algorithm

The procedure considers each of these subclasses of extra word error equally likely, and attempts to insert one of them into a grammatical sentence. Adjectives (e.g. *the great great man*) are not considered for duplication because, as with their omission, their repetition will not result in an ungrammaticality. Apart from the case of duplicate tokens, the extra words are selected from a list of tagged words compiled from a random subset of the British National Corpus [7]. This random subset contains approximately 2,500 words. Again, the procedure for inserting an extra word is based on the analysis of extra word errors in the 20,000 word error corpus of Foster [18]. The algorithm is shown in Fig. 3. Note that it is always possible to generate an extra word error within a sentence because it always possible to insert an arbitrary token at a random position.

### *4.1.3 Real-Word Spelling Errors*

An error is classified as a real-word spelling error or context-sensitive spelling error if it can be corrected by a word similar to it in spelling. Two words are considered similar in spelling if the Levenshtein distance between

**Table 1** Some English Real-Word Spelling Errors

| | | | | |
|---|---|---|---|---|
| $is \leftrightarrow if$ | $is \leftrightarrow in$ | $is \leftrightarrow it$ | $is \leftrightarrow as$ | $is \leftrightarrow us$ |
| $is \leftrightarrow its$ | $is \leftrightarrow his$ | $if \leftrightarrow in$ | $if \leftrightarrow it$ | $if \leftrightarrow of$ |
| $in \leftrightarrow it$ | $in \leftrightarrow an$ | $in \leftrightarrow on$ | $it \leftrightarrow its$ | $it \leftrightarrow at$ |

**introduce_real_word_spell_error(s)**
$s\_ungram \leftarrow \epsilon$
$eng\_spell\_list \leftarrow$ list of English real word spelling errors
$candidate\_list \leftarrow <>$
**for** each word, $w$, at position, $p$, in $s$ **do**
  **if** $< w, \_ > \in eng\_spell\_list$ **then**
    $candidate\_list \leftarrow candidate\_list + < w, p >$
**if** $candidate\_list$ is not empty **then**
  $n \leftarrow$ random integer in range $1...length(candidate\_list)$
  $< w', p' > \leftarrow candidate\_list[n]$
  $rep\_list \leftarrow < r | < w', r > \in eng\_spell\_list >$
  $k \leftarrow$ random integer in range $1...length(rep\_list)$
  $r' \leftarrow rep\_list[k]$
  $s\_ungram \leftarrow s$ with $word@p'$ replaced by $r'$
return $s\_ungram$

**Fig. 4** Real-Word Spelling Error Creation Algorithm

them is one (e.g. *to* and *too*) ([18]). Again following the error analysis carried out by Foster [18], a list of candidate English real-word spelling errors is compiled. The error creation procedure searches for all words in the input sentence which can be replaced by a word similar in spelling (subject to the pre-compiled list): one of these is then randomly selected and replaced. The list of real-word spelling errors contains 113 pairs and a sample of 15 involving function words related to the words *is*, *it*, *in* and *if* are shown in Table 1. The list contains very common English words such as *a*, *the* and *he*, and an ungrammatical sentence can be generated from most sentences. The algorithm for inserting real-word spelling errors is shown in Fig. 4.

### *4.1.4 Agreement Errors*

Subject-verb and determiner-noun number agreement errors are introduced into well-formed sentences by replacing a singular determiner, noun or verb with its plural counterpart, or vice versa. For English, subject-verb agreement errors can only be introduced for present tense verbs, and determiner-noun agreement errors can only be introduced for determiners which are marked for number, e.g. demonstratives and the indefinite article. The procedure would be more productive if applied to a morphologically richer language. According to the error analysis carried out by Foster [18], the erroneous word within an agreement error is more likely to be the rightmost word, i.e. the verb in a subject-verb agreement error or the noun in a determiner-noun agreement error. This is reflected in the algorithm for creating agreement errors which is shown in Fig. 5.

**insert_agree_error(s)**
$s\_ungram \leftarrow \epsilon$
**while** error has not been introduced and not all words in $s$ tried **do**
  $p \leftarrow$ random integer in the range $1...length(s)$
  **if** $POS(word@p) =$ noun and $POS(word@p+1) =$ number marked verb **then**
    $n \leftarrow$ random integer in the range 1...3
    **if** $n = 1$ **then**
      $s\_ungram \leftarrow s$ with $word@p$ replaced by opposite number form
    **else**
      $s\_ungram \leftarrow s$ with $word@p + 1$ replaced by opposite number form
  **else if** $POS(word@p) =$ number marked determiner and $POS(word@p + 1) =$ noun **then**
    $n \leftarrow$ random integer in the range 1...3
    **if** $n = 1$ **then**
      $s\_ungram \leftarrow s$ with $word@p$ replaced by opposite number form
    **else**
      $s\_ungram \leftarrow s$ with $word@p + 1$ replaced by opposite number form
  **else if** $POS(word@p) =$ number marked determiner and $POS(word@p+1) =$ adj and $POS(word@p+2) =$ noun **then**
    $n \leftarrow$ random integer in the range 1...3
    **if** $n = 1$ **then**
      $s\_ungram \leftarrow s$ with $word@p$ replaced by opposite number form
    **else**
      $s\_ungram \leftarrow s$ with $word@p + 2$ replaced by opposite number form
  **else if** $POS(word@p) =$ number marked verb **then**
    $s\_ungram \leftarrow s$ with $word@p$ replaced by opposite number form
return $s\_ungram$

**Fig. 5** Agreement Error Creation Algorithm

### *4.1.5 Verb Form Error*

Verb form errors are introduced into well-formed sentences by changing the tense of a verb within the sentence, e.g. changing from the present participle verb form *laughing* to the infinitival form *laugh*. The procedure for inserting this type of error proceeds by identifying all verbs within a sentence, selecting one of these verbs at random, and replacing it with another verb form, also chosen at random. The algorithm is shown in Fig. 6. Note that some transformations are not carried out, e.g. a present tense verb is not converted to its past form because the transformation will not result in an ill-formed sentence (*They laugh* versus *They laughed*) and a present tense verb is not converted to its plural or singular counterpart since this is already covered by the agreement error creation module.

### *4.1.6 Covert Errors*

James [24] uses the term *covert error* to describe a genuine language error which results in a sentence which is syntactically well-formed under some interpretation different from the intended one. The tendency of the er-

**insert_verb_form_error(s)**
*s_ungram* ← $\epsilon$
*candidate_verbs* ←<>
**for** each word, $w$, at position, $p$, in $s$ **do**
**if** POS(w) = suitable verb form **then**
*candidate_verbs* ← *candidate_verbs*+ $< w, p >$
$n$ ← random integer in range 1...*length*(*candidate_verbs*)
$< v, p' >$ ← *candidate_verbs*[$n$]
**if** $v$ is a past participle form **then**
$k$ ← random integer in range 1...3
**if** k=1 **then**
*s_ungram* ← s with *word*@$p'$ replaced by inf form
**else if** k=2 **then**
*s_ungram* ← s with *word*@$p'$ replaced by present participle form
**else**
*s_ungram* ← s with *word*@$p'$ replaced by 3rd pers sing form
**else if** $v$ is an infinitival form **then**
$k$ ← random integer in range 1...3
**if** k=1 **then**
*s_ungram* ← s with *word*@$p'$ replaced by past participle form
**else if** k=2 **then**
*s_ungram* ← s with *word*@$p'$ replaced by present participle form
**else**
*s_ungram* ← s with *word*@$p'$ replaced by 3rd pers sing form
**else if** $v$ is a present participle form **then**
$k$ ← random integer in range 1...3
**if** k=1 **then**
*s_ungram* ← s with *word*@$p'$ replaced by past participle form
**else if** k=2 **then**
*s_ungram* ← s with *word*@$p'$ replaced by infinitival form
**else**
*s_ungram* ← s with *word*@$p'$ replaced by 3rd pers sing form
**else if** $v$ is a present tense form **then**
*s_ungram* ← s with *word*@$p'$ replaced by present participle form
return *s_ungram*

**Fig. 6** Verb Form Creation Algorithm

ror creation procedure to produce covert errors was estimated by carrying out the following small experiment: sentences were randomly extracted from the BNC and the error creation procedure applied to them. 500 of the resulting sentences (the first 100 for each error type) were then manually inspected to see if the sentence structures were grammatical. The percentage of grammatical structures that are inadvertently produced for each error type and an example of each one are shown below:

(11)Agreement Errors, 7%
*Mary's staff **include** Jones,Smith and Murphy* → *Mary's staff **includes** Jones,Smith and Murphy*
(12)Real-Word Spelling Errors, 10%
*And **then**?* → *And **them**?*
(13)Missing Word Errors, 13%[2]
*She steered **Melissa** round a corner* → *She steered round a corner*
(14)Extra Word Errors, 5%
*She made no effort to check her tears* → *She made no effort to check **in** her tears*
(15)Verb Form Errors, 6%
*There was no **turning** back* → *There was no **turned** back*

The occurrence of these grammatical sentences in the artificial error corpus can be reduced by fine-tuning the error creation procedure or by using a finely grained part-of-speech tagset to tag the input corpus. For example, if the tagset could discriminate singular nouns like *staff* and *company* which can have a distributive reading from singular nouns such as *car* which don't, examples like (11) would not be produced. [3] Similarly, if verb subcategorization frames were available to the error creation procedure, it would know that the verb *steer* can be used intransitively (13) or that the verb *check* can be used with the preposition *in* (14). It is unrealistic to assume, however, that covert errors can ever be completely eliminated. They are a natural linguistic phenomenon which occur in manually created error corpora containing real errors. One could argue, therefore, that they should not be eliminated. Ideally, a probabilistic parser should be sophisticated enough to favour an ill-formed structure with a plausible reading over a well-formed structure with an unlikely reading. For example, a parser that suspects that a verb form error has occurred in the right-hand sentence of Example (15) and interprets the *turned* as a present participle verb form and *back* as an adverb is more useful than a parser which interprets *no turned back* as a noun phrase.

### *4.1.7 Iteratively Applying the Error Creation Procedure*

The output of the error creation procedure is a tagged corpus of ungrammatical sentences (including the covert errors discussed in the previous section). This corpus can then be passed as input to the procedure to create a second corpus with even noisier data. This second corpus will contain sentences containing two separate errors such as (16), or sentences such as (17) or (18) which are correctable by applying a combination of the basic *insert*, *delete*, *substitute* correction operators.

(16)*This roadmap for the **project** has been derived.* → *This roadmap for the has been derived **derived**.*
(17)*I have problems **to get** the script to run.* → *I have problems **getting** the script to run.*
(18)*What does **the thing** do?* → *What does **thing the** do?*

[2] Smith and Eisner [39,40] also note the propensity of a word omission to result in a well-formed syntactic structure.
[3] Note that example (11) is not really a covert error because the two sentences have the same meaning.

Increasingly noisy corpora can be created by iteratively applying the error creation procedure to its own output. However, if the data is too noisy it will become very difficult for a parser to accurately parse it, just as it becomes very difficult for people to understand extremely ungrammatical sentences such as (19). Is it reasonable to expect a computer parser to handle language that is problematic for the human parsing mechanism?

(19) *Hotkey Utility show the indicators on your display and save brightness adjustment each power supplying conditions.*

## 4.2 Gold Standard Transformation

The gold standard transformation procedure takes an ungrammatical sentence and a gold standard syntactic analysis of the grammatical sentence from which the ungrammatical one has been generated, and outputs a gold standard syntactic analysis of the ungrammatical sentence. The transformation method is based on three assumptions, the third assumption following on from the first two:

1. At the heart of every ungrammatical sentence, there is a grammatical sentence which expresses the same "intended" meaning as the ungrammatical sentence.
2. The role of a parser is to produce an analysis for a sentence which reflects, to a certain extent, that sentence's "intended" meaning.
3. A parser which aims to be robust to errors should produce an analysis for an ungrammatical sentence which is as close as possible to the analysis it produces for the corresponding grammatical sentence, i.e. for the grammatical sentence at the heart of the ungrammatical sentence.

In keeping with these assumptions, the transformation procedure operates by changing as little as possible in the original grammatical sentence analysis to produce the analysis of the ungrammatical sentence. Ungrammatical treebanks can be automatically generated from any type of treebank, regardless of the syntactic annotation scheme it employs. However, in this article, attention is restricted to context-free phrase structure trees. Examples are provided for the error types described in Section 4.1.

### *4.2.1 Real-Word Spelling Errors, Agreement Errors and Verb Form Errors*

Consider the grammatical sentence (20) and the ungrammatical sentence (21) which contains a real-word spelling error:

(20) *A romance **is** coming your way.*
(21) *A romance **in** coming your way.*

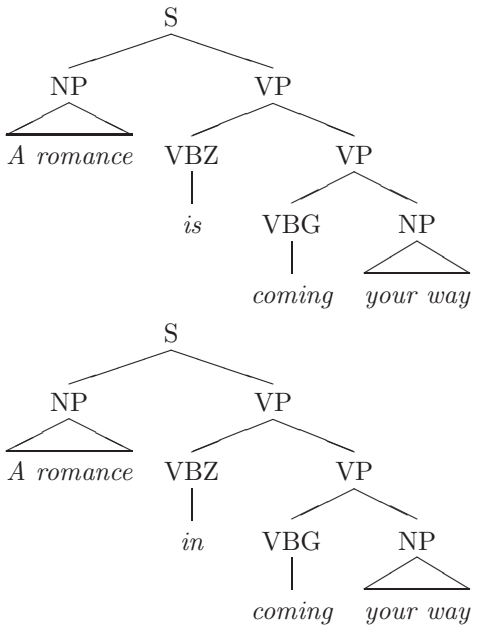


**Fig. 7** Gold Standard Parses for Sentences (20) and (21)

Fig. 7 depicts a Penn-Treebank-style gold standard parse tree[4] for the grammatical sentence (20) and, underneath it, the parse tree which will be produced by the transformation procedure for the ungrammatical sentence (21). This is considered to be the gold standard parse for the ungrammatical sentence because it makes the crucial recognition that the word *in* is part of a verb phrase and contrasts in this way with another parse for the same sentence, shown in Fig. 8, in which the sequence *in coming your way* is analysed as a prepositional phrase. A parser which produces the parse in Fig. 7 is robust to errors since it is able to see right through an ungrammatical sentence to the grammatical sentence at its heart, and produce a parse which reflects the meaning of the grammatical sentence.[5] The example sentence (21) contains a real-word spelling error but the same transformation would apply to any error correctable by a substitution, e.g. an agreement or a verb form error. This transformation is the substitution of the erroneous word onto the original word in the original analysis.

### *4.2.2 Missing Word Errors*

Consider the grammatical sentence (22) and its ungrammatical counterpart (23):

(22) *Prices are expected **to drop**.*

[4] Penn-II functional tags and traces have been omitted, since they are not needed to explain the tree transformations.

[5] Note that the part-of-speech of the word *in* in the gold standard parse tree is `VBZ` rather than `IN`. A similar decision was made in the annotation of typos in the Switchboard Corpus (`http://www.cis.upenn.edu/~bies/manuals/tagguid2.pdf`)

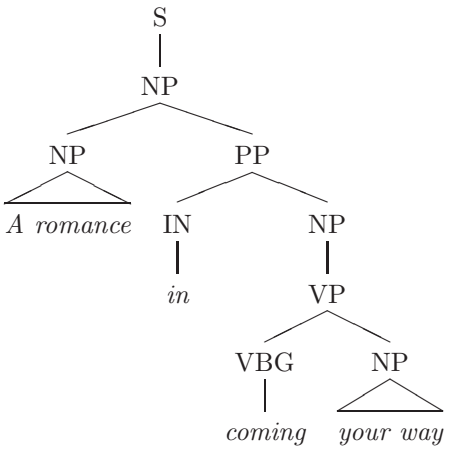


**Fig. 8** A Suboptimal Parse Tree for Sentence (21)

(23)*Prices are expected* ***drop****.*

A gold standard parse tree for the grammatical (22) is shown in Fig. 9, with the gold standard parse tree which will be automatically generated for the ungrammatical (23) underneath. The bottom tree is produced by replacing the pre-terminal category *(TO to)* in the top tree in Fig. 9 with the trace *(-NONE- 0)*. In contrast, Fig. 10 shows a less accurate parse tree for Sentence (23).

### *4.2.3 Extra Word Errors*

Consider the grammatical sentence (24) and the ungrammatical sentence (25) which contains an unnecessary extra word *to*:

(24)*Annotators parse the sentences.*
(25)*Annotators parse* ***to*** *the sentences.*

Fig. 11 shows the gold standard parse tree for the grammatical (24), along with the *two* gold standard parse trees which will be generated automatically by the transformation procedure for the ungrammatical (25). In the ungrammatical gold standard trees, the superfluous *to* does not affect the constituent structure of the sentence (above the pre-terminal level). The only difference between the two trees is the level where the word *to* is attached. In both, *to* has not introduced any extra structure, which is a desirable result since the word does not contribute to the sentence's meaning. Contrast this with the parse tree in Fig. 12, in which the presence of the word *to* has caused a prepositional phrase to be introduced.

### *4.2.4 Tree Transformation Algorithm*

The tree transformation algorithm is shown in Fig. 13. The top-level procedure takes as input a treebank tree, $t$, and a part-of-speech tagged sentence, $s$. The sentence $s$

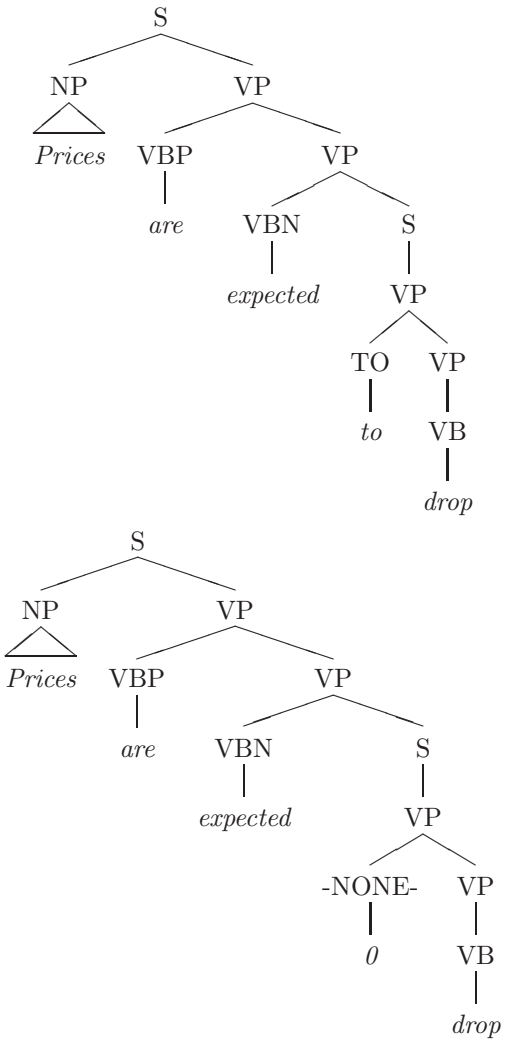


**Fig. 9** Gold Standard Parse Trees for Sentences (22) and (23)

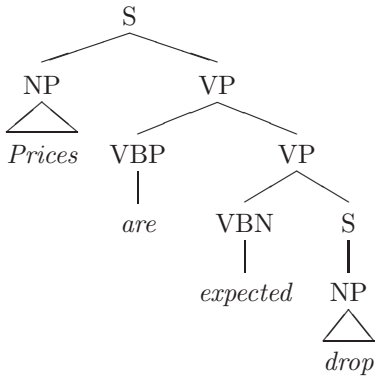


**Fig. 10** A Suboptimal Parse Tree for Sentence (23)

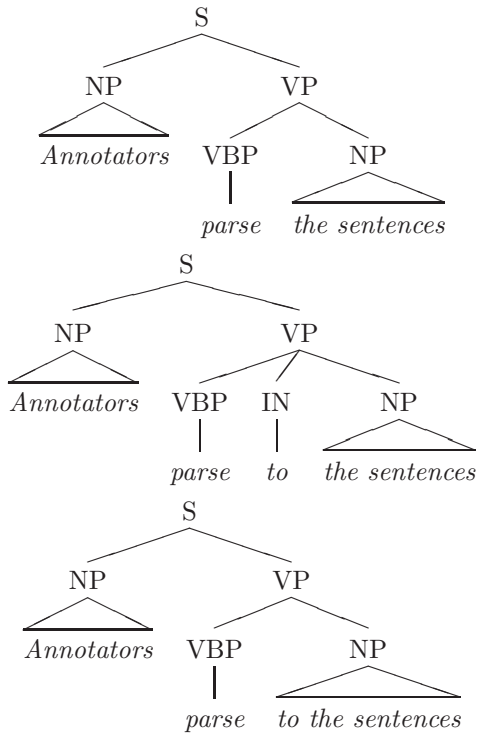


**Fig. 11** Gold Standard Parse Trees for Sentences (24) and (25)

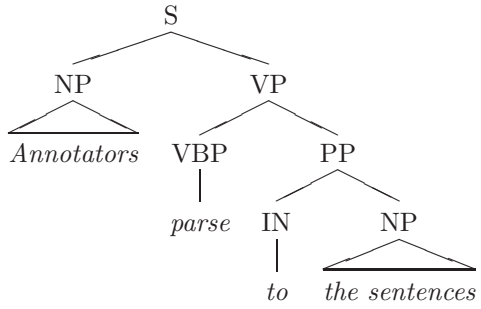


**Fig. 12** A Suboptimal Parse Tree for Sentence (25)

is the ungrammatical version of the sentence dominated by $t$. The procedure returns a list of ungrammatical trees dominating $s$. A list of trees rather than a single tree is returned because for extra word errors there can be more than one way of transforming the original tree to describe the ungrammatical sentence while preserving the intended meaning (see Fig 11). If $s$ contains a missing word error, a real-word spelling error, a verb form error or an agreement error, the list returned will always contain only one tree. Note that the algorithm assumes that the variables *term_count* and *count* are global variables available to all procedures.

**transform_tree(t,s)**
  $trees\_ungram \leftarrow <>$
  $i \leftarrow 1$
  **while** $yield_i(t) = word@i$ in $s$ **do**
    $i \leftarrow i + 1$
  **if** $error\_type(s)$ = real_word or $error\_type(s)$ = agreement
  or $error\_type(s)$ = verb_form **then**
    $term\_count = 0$
    $trees\_ungram \leftarrow trees\_ungram + replace(t, word@i, i)$
  **else if** $error\_type(s)$ = missing **then**
    $term\_count = 0$
    $trees\_ungram \leftarrow trees\_ungram + delete(t, i)$
  **else if** $error\_type(s)$ = extra **then**
    create new tree, $t'$ of depth 1
    $root(t') \leftarrow POS(word@i)$ in $s$
    $yield_1(t') \leftarrow word@i$ in $s$
    $trees\_no \leftarrow$ number of ancestors, $t''$, of $t$, such that $root(t'')$ is not a pre-terminal and $yield_1(t'') = yield_{i-1}(t)$ or $yield_1(t'') = yield_i(t)$
    $trees\_count \leftarrow 0$
    **while** $trees\_count < trees\_no$ **do**
      $term\_count = 0$
      $count = 0$
      $trees\_ungram \leftarrow$
      $trees\_ungram + add(t, t', i, trees\_count, 0)$
      $trees\_count \leftarrow trees\_count + 1$
  return $trees\_ungram$

**replace(tree,word,pos)**
  **if** $root(tree)$ is a pre-terminal **then**
    $term\_count \leftarrow term\_count + 1$
    **if** $term\_count = pos$ **then**
      $yield_1(tree) \leftarrow word$
  **else**
    **for** all daughters, $d$, of $tree$ **do**
      $d \leftarrow replace(d, word, pos)$
  return $tree$

**delete(tree,pos)**
  **if** $root(tree)$ is a pre-terminal **then**
    $term\_count \leftarrow term\_count + 1$
    **if** $term\_count = pos$ **then**
      $yield_1(tree) \leftarrow 0$
      $root(tree) \leftarrow -NONE-$
  **else**
    **for** all daughters, $d$, of $tree$ **do**
      $d \leftarrow delete(d, pos)$
  return $tree$

**add(tree,new,pos,limit)**
  **if** $root(tree)$ is a pre-terminal **then**
    $term\_count \leftarrow term\_count + 1$
  **else**
    **if** $pos > term\_count$ and
    $pos <= term\_count + length(yield(tree))$ **then**
      $count \leftarrow count + 1$
      **if** $count = limit$ **then**
        $daughter_{(pos-term\_count)}(tree) \leftarrow new$
    **else**
      **for** all daughters, $d$, of $tree$ **do**
        $d \leftarrow add(d, new, pos, limit)$
    return $tree$

**Fig. 13** Tree Transformation Algorithm

#### *4.2.5 Iteratively Applying the Gold Standard Transformation Procedure*

Just as the error creation procedure can be applied to its own output to create increasing levels of grammatical noise, the gold standard transformation algorithm can take as input a tree that it has already undergone transformation and produce another tree as output. Consider, for example, the ungrammatical sentence (21) repeated as (26) and sentences (27), (28) and (29) which are the result of introducing a second error into sentence (26):

(26) *A romance* ***in*** *coming your way.*
(27) *A* ***romances in*** *coming your way.*
(28) *A romance* ***in*** *coming way.*
(29) *A* ***the*** *romance* ***in*** *coming your way.*

In order to produce gold standard parse trees for sentences (27), (28) and (29), the input to the tree transformation procedure is the bottom tree in Fig. 7. The output trees are shown in Fig. 14.

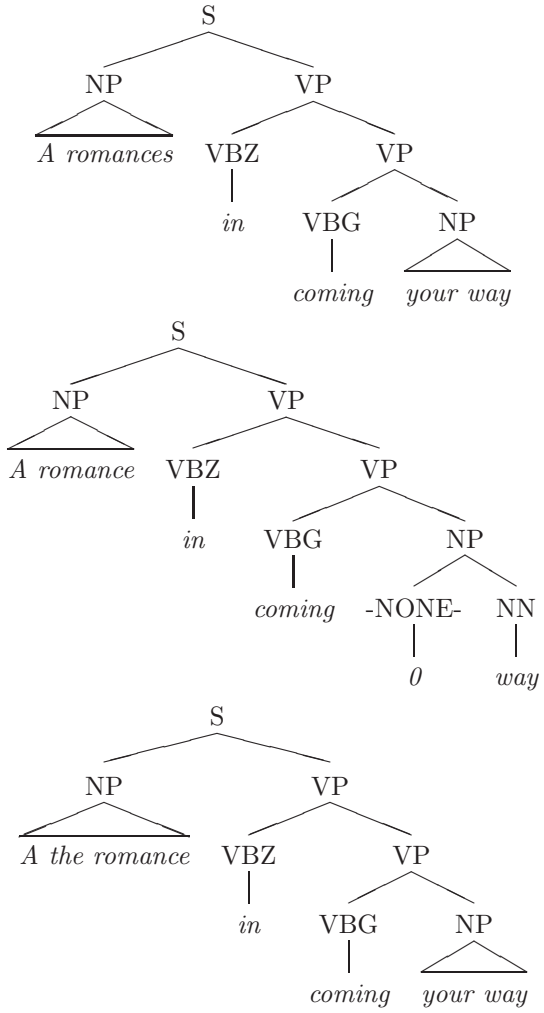


**Fig. 14** Gold Standard Parses for Sentences (27), (28) and (29)

## 5 Experiments with an Ungrammatical Penn Treebank

In this section, the usefulness of an automatically created ungrammatical treebank is demonstrated by describing a parser evaluation experiment and a parser retraining experiment which are carried out using an ungrammatical version of the Wall Street Journal portion of the Penn Treebank [32,31]. The evaluation experiment is described in Section 5.1 and the retraining experiment in Section 5.2.

### 5.1 Parser Evaluation

The aim of this experiment is to evaluate how well two lexicalized, history-based, generative, statistical parsers cope with errors in text: a parser that copes well with errors produces, for an ungrammatical sentence, an analysis which closely resembles the analysis it would produce for the sentence without the error.

Section 5.1.1 contains a description of how the experiment was carried out and Section 5.1.2 presents the results, which are then discussed in Section 5.1.3.

#### *5.1.1 Method*

The error creation procedure described in Section 4.1 is applied to the 2416 sentences in Section 23 of the WSJ portion of the Penn Treebank [32,31], resulting in an error corpus of 2416 sentences (133 sentences containing a verb form error, 234 sentences containing an agreement error, 511 sentences containing a real-word spelling error, 613 sentences containing an extra word and 925 sentences with a missing word). The gold standard transformation procedure described in Section 4.2 is then applied, resulting in an ungrammatical version of WSJ section 23. A second noisier test set is created by applying the error creation and gold standard transformation procedures to the first ungrammatical version of WSJ section 23 (see Sections 4.1.7 and 4.2.5).

Two statistical parsers trained on the original grammatical WSJ sections 2-21 are used to parse the ungrammatical sentences. The first parser is Bikel's implementation of Collins' generative head-driven probabilistic Model 2 [5,12]. The second parser is the June 2006 version of Charniak and Johnson's two-stage parser [9]. The first-stage is a lexicalized, generative, probabilistic parser [8] and the second stage is a maximum entropy reranker which exploits features of the entire parse tree to reorder the n-best parse trees produced by the first stage parser [27,11,9]. For this experiment, the input to both parsers is untagged.

**Table 2** Parsers on Ungrammatical WSJ23

| | P | R | F |
|---|---|---|---|
| **Bikel** | | | |
| Grammatical | 85.9 | 85.8 | 85.9 |
| Ungrammatical_1 | 81.3 | 80.7 | 81.0 |
| Ungrammatical_2 | 76.4 | 75.4 | 75.9 |
| **Charniak & Johnson** | | | |
| Grammatical | 91.7 | 90.8 | 91.3 |
| Ungrammatical_1 | 87.4 | 85.6 | 86.5 |
| Ungrammatical_2 | 83.2 | 80.7 | 81.9 |

**Table 3** Parsers on Ungrammatical WSJ23: Breakdown by Error Type

| Error Type | P | R | F |
|---|---|---|---|
| **Bikel** | | | |
| Missing Word | 82.8 | 79.2 | 81.0 |
| Extra Word | 80.3 | 84.0 | 82.1 |
| Real-Word Spelling | 79.6 | 79.5 | 79.6 |
| Agreement | 83.2 | 82.7 | 83.0 |
| Verb Form | 79.6 | 78.5 | 79.0 |
| **Charniak & Johnson** | | | |
| Missing Word | 88.7 | 83.6 | 86.1 |
| Extra Word | 87.0 | 89.2 | 88.1 |
| Real-Word Spelling | 84.2 | 83.2 | 83.7 |
| Agreement | 90.8 | 89.2 | 90.0 |
| Verb Form | 88.3 | 86.5 | 87.4 |

The parses produced by both parsers are evaluated against the ungrammatical gold standard WSJ23 parses using the Parseval [6] labelled precision/recall metric. According to this metric, a constituent in a test parse tree is considered to be correct if it spans the same sequence of words and has the same label as a constituent in the corresponding gold standard parse tree. Precision represents the number of correct constituents divided by the total number of constituents produced by the parser. Recall represents the number of correct constituents divided by the total number of constituents in the gold standard set. The f-score is the harmonic mean of precision and recall. In the case of extra word errors, there is potentially more than one gold standard analysis for each sentence (see, for example, Fig 11), and therefore the test sentence parse is evaluated against each of its gold standard parses, and the highest f-score is chosen.

*5.1.2 Results*

Table 2 shows labelled precision, recall and f-score results calculated by evaluating the two parsers against the two ungrammatical versions of WSJ23 using the Parseval measures. "Ungrammatical_1" is the test set produced by applying the error creation and gold standard transformation procedures to the original grammatical WSJ23. "Ungrammatical_2" is the test set produced by applying the error creation and gold standard transformation procedures to Ungrammatical_1. The first row in Table 2 indicates the scores received by the parsers on the original grammatical WSJ23 sentences. The first row figures represent an upper bound for the ungrammatical sentence results, because, as was illustrated in Section 4.2, the grammatical and ungrammatical gold standard trees are isomorphic above the pre-terminal level and pre-terminal constituents or part-of-speech tags are ignored in calculation of constituent accuracy. Table 3 gives a breakdown of the Ungrammatical_1 results by error type.

*5.1.3 Discussion*

As might be expected, the presence of a grammatical error in a sentence has an adverse effect on both parsers. The effect on both is quite similar, with an absolute f-score deterioration of 4.9% on Ungrammatical_1 for Bikel's parser, and an absolute f-score deterioration of 4.8% for Charniak and Johnson's parser. For both parsers, there is an even larger deterioration for Ungrammatical_2, 10% for Bikel's parser and 9.4% for Charniak and Johnson's parser. Again, this is unsurprising, since Ungrammatical_2 contains noisier sentences than the sentences in Ungrammatical_1: each has an edit distance of two from the original grammatical WSJ23 sentences.

The results in Tables 2 and 3 show that ungrammatical sentences containing agreement errors achieve scores which are the closest to the upper bound, suggesting that this type of error does not generally distract these parsers from finding the correct analysis. For both parsers, there is no significant difference between the results for subject-verb agreeement errors and those for determiner-noun agreement errors. Real-word spelling errors are a problem for both parsers. It is not surprising that this error type performs badly, since the replacing word and the replaced word often have little in common grammatically. The worst performing error type for Bikel's parser is the wrong verb form error. Interestingly, this is not the case for Charniak and Johnson's parser. In particular, Charniak and Johnson's parser is more robust than Bikel's parser to verb form errors involving the conversion of either an infinitival or present tense indicative verb form to a present participle form. Examples are shown in (30) and (31):

(30) infinitival → present participle
Charniak and Johnson:
*(S (NP Hooker's philosophy) (VP was (S (VP to (VP **building** and sell)))))*
Bikel:
*(S (NP Hooker's philosophy) (VP was (PP to (NP **building** and sell))))*

(31) present indicative → present participle
Charniak and Johnson:
*(S (NP That figure) (VP **climbing** (PP to (NP about 47%))))*
Bikel:
*(S (NP That) (VP figure (VP **climbing** (PP to (NP about 47%)))))*

For both parsers, extra word errors achieve a higher recall score in comparison to their precision score which suggests that this kind of error tends to introduce unwanted structure into a parse. The two parsers' scores for extra word errors are slightly lower when the extra word is a random token inserted at an arbitrary point in the sentence, as opposed to extra word errors involving adjacent duplicate tags or tokens.

For both parsers and in contrast to extra word errors, missing word errors achieve a higher precision score in comparison to their recall score, suggesting that a lack of relevant structure is associated with this kind of error. This is what one might expect. Analysing the missing word errors by the part of speech that is omitted, it is clear that both parsers cope well with the omission of determiners and nouns, and less well with the omission of verbs, prepositions and conjunctions. Of all the error types, the missing word error is most associated with covert errors (see Section 4.1.6). A likely explanation for the better performance of errors involving missing nouns is that their omission is more inclined to result in a grammatical structure, e.g. in a noun-noun compound or as objects to verbs with both transitive and intransitive uses (see Example (13)). An error involving a missing determiner is likely to be contained locally within a noun phrase and less likely to affect the parsing of other constituents in a sentence, as would appear to be the case for missing verbs, prepositions and conjunctions.

## 5.2 Parser Retraining

The aim of this experiment is to determine the effect of training the two parsers evaluated in Section 5.1 on an ungrammatical version of the WSJ. It is expected that this will have a positive effect on the parsers' performance on the ungrammatical WSJ Section 23 sentences. Section 5.2.1 contains a description of how the experiment was carried out, Section 5.2.2 presents the results, and these results are discussed in Section 5.2.3.

### *5.2.1 Method*

The error creation and tree transformation procedures described in Section 4 are applied to Sections 2-21 of the WSJ portion of the Penn Treebank. Following the error analysis carried out by Foster[18], 4 of the 20 training sections (Sections 2-5) contain sentences with more than one error or an error correctable by two applications of the *insert*, *delete* or *substitute* correction operators. This is achieved by applying the error creation and tree transformation procedures to Sections 2-5 and then applying the procedures again to their own output (see Sections 4.1.7 and 4.2.5). The parsers evaluated in Section 5.1 are trained on this new training set. In a second experiment, the ungrammatical training set is combined with the original grammatical WSJ sections 2-21, and the two parsers are trained on this combined set.

### *5.2.2 Results*

The overall results for the parser retraining experiments are shown in Table 4. The first column of results represent the scores for both parsers on the three test sets before the retraining and are a repeat of the results shown in Table 2. The middle column of results are those achieved after the parsers have been retrained on the ungrammatical WSJ2-21 only, and the third set of results are those achieved when the parsers are retrained on a combination of the ungrammatical WSJ2-21 and the original WSJ2-21. A breakdown for individual error types is shown in Table 5.

### *5.2.3 Discussion*

As expected, both parsers achieve an improved f-score on the ungrammatical test data, when trained on ungrammatical data alone. For the Ungrammatical_1 test set, Bikel's parser achieves an absolute improvement of 1.2%, and Charniak and Johnson's parser achieves an absolute improvement of 1.4%. The improvement is statistically significant for both Bikel's parser ($p < 0.0001$ for precision and $p < 0.004$ for recall) and Charniak and Johnson's parser ($p < 0.0001$ for precision and recall).[6] For the noiser Ungrammatical_2 test set, the improvements are even greater, 3.9% for Bikel's parser and 3.7% for Charniak and Johnson's parser. Both improvements are statistically significant ($p < 0.0001$). Unsurprisingly, when trained on ungrammatical data alone, the performance of both parsers on grammatical sentences is negatively affected. There is an absolute deterioration of 1.7% for Bikel's parser and an absolute deterioration of 1.2% for Charniak and Johnson's parser. The deterioration is statistically significant for both parsers ($p < 0.0001$).

The situation improves (although not quite as much as hoped) when the parsers are trained on both the grammatical and ungrammatical versions of WSJ2-21. For Ungrammatical_1, Bikel's parser achieves a statistically significant ($p < 0.0001$) improvement of 1.7% over the baseline of training on the original grammatical WSJ2-21. Charniak and Johnson's parser achieves an improvement of 1.6%, also statistically significant ($p < 0.0001$). For Ungrammatical_2, there is an improvement of 4.3% for Bikel's parser and 4.0% for Charniak and Johnson's parser, both improvements statistically significant ($p < 0.0001$). For the original WSJ23, the results

[6] Statistical significance is determined using a "stratified shuffling" method which repeatedly shuffles sentence scores between two sets of evaluation results for two parser models in order to test the null hypothesis that the two models are the same. After every shuffle, the difference between the two result sets is calculated and a count is incremented if the difference is greater or equal to the original observed difference. After 10,000 iterations, the likelihood of incorrectly rejecting the null hypothesis is (count+1)/10001. The software used to perform the test was downloaded from `http://www.cis.upenn.edu/\%7Edbikel/software.html`.

**Table 4** Retraining Results

| | P | R | F | P | R | F | P | R | F |
|---|---|---|---|---|---|---|---|---|---|
| | Gram Only | | | Ungram Only | | | Gram and Ungram | | |
| **Bikel** | | | | | | | | | |
| Grammatical | 85.9 | 85.8 | 85.9 | 84.3 | 82.2 | 84.2 | 85.1 | 85.0 | 85.1 |
| Ungrammatical_1 | 81.3 | 80.7 | 81.0 | 82.7 | 81.6 | 82.2 | 83.2 | 82.3 | 82.7 |
| Ungrammatical_2 | 76.4 | 75.4 | 75.9 | 80.7 | 79.0 | 79.8 | 81.0 | 79.4 | 80.2 |
| **Charniak & Johnson** | | | | | | | | | |
| Grammatical | 91.7 | 90.8 | 91.3 | 90.7 | 89.5 | 90.1 | 91.3 | 90.2 | 90.8 |
| Ungrammatical_1 | 87.4 | 85.6 | 86.5 | 89.1 | 86.8 | 87.9 | 89.3 | 87.0 | 88.1 |
| Ungrammatical_2 | 83.2 | 80.7 | 81.9 | 87.4 | 83.9 | 85.6 | 87.6 | 84.2 | 85.9 |

**Table 5** Retraining Results: (Breakdown by Error Type)

| **Error Type** | **P** | **R** | **F** | **P** | **R** | **F** | **P** | **R** | **F** |
|---|---|---|---|---|---|---|---|---|---|
| | **Gram Only** | | | **Ungram Only** | | | **Gram and Ungram** | | |
| **Bikel** | | | | | | | | | |
| Missing Word | 82.8 | 79.2 | 81.0 | 83.0 | 79.7 | 81.3 | 83.3 | 80.1 | 81.7 |
| Extra Word | 80.3 | 84.0 | 82.1 | 82.4 | 83.7 | 83.0 | 83.1 | 84.6 | 83.8 |
| Real-Word Spelling | 79.6 | 79.5 | 79.6 | 83.6 | 83.4 | 83.5 | 83.3 | 83.8 | 83.8 |
| Agreement | 83.2 | 82.7 | 83.0 | 81.9 | 81.3 | 81.6 | 82.3 | 81.9 | 82.1 |
| Verb Form | 79.6 | 78.5 | 79.0 | 82.1 | 80.9 | 81.5 | 83.3 | 82.2 | 82.7 |
| **Charniak & Johnson** | | | | | | | | | |
| Missing Word | 88.7 | 83.6 | 86.1 | 88.9 | 84.1 | 86.4 | 88.9 | 84.2 | 86.5 |
| Extra Word | 87.0 | 89.2 | 88.1 | 89.0 | 88.9 | 89.0 | 89.0 | 89.2 | 89.1 |
| Real-Word Spelling | 84.2 | 83.2 | 83.7 | 90.2 | 89.1 | 89.7 | 90.3 | 89.2 | 89.7 |
| Agreement | 90.8 | 89.2 | 90.0 | 89.5 | 88.2 | 88.9 | 90.0 | 88.5 | 89.3 |
| Verb Form | 88.3 | 86.5 | 87.4 | 87.3 | 85.9 | 86.6 | 89.0 | 87.4 | 88.2 |

for both parsers still, unfortunately, represent a decrease from the baseline f-score results: -0.8% for Bikel's parser and -0.5% for Charniak and Johnson's parser. Although these are small differences, they are statistically significant ($p < 0.0001$ for recall and $p < 0.03$ for precision for Charniak and Johnson's parser, $p < 0.0001$ for precision and recall for Bikel's parser). Training on a combination of grammatical and ungrammatical data seems to be better than training on ungrammatical data alone but it is clear from these results that a more sophisticated approach to combining both types of training data is required. An attempt was made to weight the training material in favour of the grammatical sentences by using ten copies of the original WSJ2-21 sentences, but this did not improve results.

The breakdown by error type shows very similar trends for both parsers. Comparing the first and third columns in Table 5, we can see that the most pronounced improvement is shown for real-word spelling errors, with Bikel's parser achieving an absolute f-score increase of 4.2% and Charniak and Johnson's parser achieving an absolute f-score increase of 6.0%. The parsers have learnt new part-of-speech tags for certain words and this knowledge can lead to improved parse trees for ungrammatical input. (33) is an example: the knowledge that *where* can sometimes be confused with the verb *were* has allowed Charniak and Johnson's parser to correctly posit a verb phrase, where previously it did not (32):

(32) *(S But (NP There) (WHADVP **where**) (NP no buyers))*

(33) *(S But (NP There) (VP **where** (NP no buyers)))*

If the parsers were initially robust to a particular error type, the inclusion of ungrammatical sentences and their trees in the training data does not help. In fact, it causes a deterioration, just as it causes a deterioration for the grammatical sentences. For both parsers, this happens for agreement errors and for certain types of missing word errors (missing nouns and determiners). It happens to a certain extent for verb form errors with Charniak and Johnson's reranking parser, although the net effect of combining grammatical and ungrammatical parse trees in the training material is positive. The parse trees (34) and (35) are examples of how the inclusion of ungrammatical material in the training set can confuse the parsers: Charniak and Johnson's parser successfully ignores the agreement error in the sentence (34) when it is trained on grammatical data alone, but when it is trained on a mixture of grammatical sentences and various kinds of ungrammatical sentences, it finds the analysis (35) in which *allocates* is a noun and a verb has been omitted more probable than the one in which it is a singular or plural verb.

(34) *(S (NP The **negotiations**) (VP **allocates** (NP about 15%) (PP to (NP foreign suppliers))))*

(35) *(S (NP The **negotiations allocates**) (VP (NP about 15%) (PP to (NP foreign suppliers))))*

## 6 Concluding Remarks

This article has introduced the concept of an automatically generated treebank of ungrammatical sentences. The purpose of such a treebank is to provide a source of ungrammatical test and training data for parsers. The treebank creation procedure involves the automatic introduction of common grammatical errors into the sentences in an existing treebank and the automatic transformation of the treebank analyses into analyses of the automatically generated ungrammatical sentences. The basic idea is that the original analysis is changed in as *minimal* a way as possible, so that the same semantic representation can be derived from an ungrammatical sentence parse as from its grammatical counterpart. The research described in this article has focused on five grammatical error types which occur frequently in written text: missing word errors, extra word errors, real-word spelling errors, agreement errors and verb form errors. By applying the error creation and tree transformation to its own output, errors involving some combination of the above five error types can also be handled.

This article described two experiments which employed an ungrammatical version of the Wall Street Journal section of the Penn Treebank. The first experiment was a parser evaluation experiment which tested the performance of two state-of-the-art WSJ-trained statistical parsers on sentences containing ungrammatical data. The Parseval metric was used to perform the evaluation, but the experiment could be repeated using another constituency-based metric such as the Leaf Ancestor metric [37], or by transforming the constituent analyses into dependency analyses [26]. The evaluation experiment showed that both parsers are fairly robust to grammatical noise, in particular to sentences containing agreement errors. The second experiment tested these two parsers on the same ungrammatical data, after retraining them on noisy versions of WSJ2-21. When trained on ungrammatical parses alone, the parsers performed slightly better on ungrammatical data and slightly worse on grammatical data, compared to when they were trained on grammatical parses alone. Combining both sets of training data yielded small but encouraging improvements for both the grammatical and ungrammatical test data.

An immediate future aim is to see if the retrained parsers behave in the same manner when faced with *real* erroneous data. The problem here is the time it will take to annotate the erroneous data, a motivation for creating an ungrammatical treebank in the first place. Nevertheless, some evaluation on real data is necessary to confirm the results presented here. Another challenge remains to improve performance on ungrammatical data without *any* adverse effect on grammatical data and a possible way forward is a two-stage parsing model, in which the grammar derived from the ungrammatical treebank is only employed when there is reason to suggest that the input sentence may contain an error. This is a matter for further research.

## Acknowledgments

Thank you to Joachim Wagner, Josef van Genabith and Deirdre Hogan for interesting discussion and advice. I am very grateful to the reviewers for their insightful comments and helpful suggestions. I am also grateful to the IRCSET Embark Initiative Postdoctoral Fellowship Award Scheme for funding this research (P04232).

## References

1. Becker, M., Bredenkamp, A., Crysmann, B., Klein, J.: Annotation of error types for german news corpus. In: Proceedings of the ATALA Workshop on Treebanks. Paris, France (1999)
2. Bender, E.M., Flickinger, D., Oepen, S., Baldwin, T.: Arboretum: Using a precision grammar for grammar checking in CALL. In: Proceedings of the InSTIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems. Venice, Italy (2004)
3. Bigert, J.: Probabilistic detection of context-sensitive spelling errors. In: Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04), vol. Five, pp. 1633–1636. Lisbon, Portugal (2004)
4. Bigert, J., Sjöbergh, J., Knutsson, O., Sahlgren, M.: Unsupervised evaluation of parser robustness. In: Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICling-05), pp. 142–154. Mexico City, Mexico (2005)
5. Bikel, D.: On the parameter space of generative lexicalized parsing models. Ph.D. thesis, University of Pennslyvania (2004)
6. Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., Strzalkowski, T.: A procedure for quantitatively comparing the syntactic coverage of English grammars. In: Proceedings of the 1991 DARPA Speech and Natural Language Workshop, pp. 306–311 (1991)
7. Burnard, L.: User reference guide for the British National Corpus. Tech. rep., Oxford University Computing Services (2000)
8. Charniak, E.: A maximum-entropy-inspired parser. In: Proceedings of the Annual Meeting of the North American Association for Computational Linguistics (NAACL-00), pp. 132–139. Seattle, Washington (2000)
9. Charniak, E., Johnson, M.: Course-to-fine n-best-parsing and maxent discriminative reranking. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05), pp. 173–180. Ann Arbor (2005)
10. Choudhury, M., Saraf, R., Jain, V., Sarkar, S., Basu, A.: Investigation and modeling of the structure of texting language. In: Proceedings of the Workshop on Analytics for Noisy Unstructured Data (AND-07) at the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp. 63–70. Hyderabad, India (2007)
11. Collins, M.: Discriminative reranking for natural language parsing. In: Machine Learning: Proceedings of the 17th International Conference (ICML 2000), pp. 175–182. Stanford, California (2000)

12. Collins, M.: Head-driven statistical models for natural language parsing. Computational Linguistics **29**(4), 499–637 (2003)
13. Creswell, C., Schwartzmyer, N., Srihari, R.: Information extraction for multi-participant, task-oriented, synchronous, computer-mediated communication: A corpus study of chat data. In: Proceedings of the Workshop on Analytics for Noisy Unstructured Data (AND-07) at the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp. 131–138. Hyderabad, India (2007)
14. Crystal, D. (ed.): Language and the Internet. Cambridge University Press (2001)
15. Douglas, S., Dale, R.: Towards robust PATR. In: Proceedings of the 15th International Conference on Computational Linguistics (COLING-92), pp. 468–474. Nantes, France (1992)
16. Emi, I., Uchimoto, K., Isahara, H.: The overview of the SST speech corpus of Japanese Learner English and evaluation through the experiment on automatic detection of learners' errors. In: Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04), vol. Four, pp. 1435–1439. Lisbon, Portugal (2004)
17. Foster, J.: Parsing ungrammatical input: An evaluation procedure. In: Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04), vol. Six, pp. 2039–2042. Lisbon, Portugal (2004)
18. Foster, J.: Good reasons for noting bad grammar: Empirical investigations into the parsing of ungrammatical written English. Ph.D. thesis, University of Dublin, Trinity College (2005)
19. Foster, J., Vogel, C.: Good reasons for noting bad grammar: Constructing a corpus of ungrammatical language. In: S. Kepser, M. Reis (eds.) Pre-Proceedings of the International Conference on Linguistic Evidence: Empirical, Theoretical and Computational Perspectives, pp. 151–152. organized by the Sonderforschungsbereich 441 "Linguistic Data Structures", University of Tübingen, Germany, Tübingen (2004)
20. Foster, J., Vogel, C.: Parsing ill-formed text using an error grammar. Artificial Intelligence Review: Special AICS2003 Issue **21**, 269–291 (2004)
21. Fouvry, F.: Robust processing for constraint-based grammar formalisms. Ph.D. thesis, Department of Language and Linguistics, University of Essex (2003)
22. Granger, S.: International corpus of Learner English. In: J. Aarts, P. de Haan, N.Oostdijk (eds.) English Language Corpora: Design, Analysis and Exploitation, pp. 57–71. Rodopi, Amsterdam (1993)
23. Hashemi, S.S.: Ambiguity resolution by reordering rules in text containing errors. In: Proceedings of the 10th International Conference on Parsing Technologies, pp. 69–79. Prague, Czech Republic (2007)
24. James, C.: Errors in Language Learning and Use: Exploring Error Analysis. Addison Wesley Longman (1998)
25. Jensen, K., Heidorn, G., Miller, L., Ravin, Y.: Parse fitting and prose fixing: Getting a hold on ill-formedness. American Journal of Computational Linguistics **9**(3–4), 147—160 (1983)
26. Johansson, R., Nugues, P.: Extended constituent-to-dependency conversion for English. In: J. Nivre, H.J. Kaalep, K. Muischnek, M. Koit (eds.) Proceedings of NODALIDA 2007, pp. 105–112. Tartu, Estonia (2007)
27. Johnson, M., Geman, S., Canon, S., Chi, Z., Riezler, S.: Estimators for stochastic "unification-based" grammars. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99), pp. 535–541. San Francisco, California (1999)
28. Kepser, S., Steiner, I., Sternefeld, W.: Annotating and querying a treebank of suboptimal structures. In: Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories (TLT2004), pp. 63–74. Tuebingen, Germany (2004)
29. Lee, K.J., Kweon, C.J., Seo, J., Kim, G.C.: A robust parser based on syntactic information. In: Proceedings of the Seventh Conference of the European Association for Computational Linguistics (EACL-95), pp. 223–228 (1995)
30. Lopresti, D.: Performance evaluation for text processing of noisy inputs. In: Proceedings of the 20th Annual ACM Symposium on Applied Computing (Document Engineering Track), pp. 759–763. Sante Fe, New Mexico (2005)
31. Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The Penn Treebank: Annotating predicate argument structure. In: Proceedings of the 1994 ARPA Speech and Natural Language Workshop, pp. 114–119. Princeton, New Jersey (1994)
32. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics **19**(2), 313–330 (1993)
33. Menzel, W.: Robust processing of natural language. In: Proceedings of the 19th Annual German Conference on Artificial Intelligence, Lecture Notes in Computer Science 981, pp. 19–34. Bielefeld, Germany (1995)
34. Nicholls, D.: The Cambridge Learner Corpus – error coding and analysis. In: Summer Workshop on Learner Corpora. Tokyo, Japan (1999)
35. Okanohara, D., Tsujii, J.: A discriminative language model with pseudo-negative examples. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 73–80. Prague, Czech Republic (2007)
36. Penstein Rosé, C., Lavie, A.: An efficient distribution of labor in a two stage robust interpretation process. In: Proceedings of 1997 Conference on Empirical Methods in Natural Language Processing (EMNLP-97), pp. 1129–1135 (1997)
37. Sampson, G., Babarczy, A.: A test of the Leaf-Ancestor metric for parse accuracy. In: J. Carroll, A. Frank, D. Lin, D. Prescher, H. Uszkoreit (eds.) Proceedings of the "Beyond Parseval - Towards Improved Evaluation Measures for Parsing Systems" Workshop at the 3rd International Conference on Linguistic Resources and Evaluation (LREC). Las Palmas, Gran Canaria (2002)
38. Schneider, D., McCoy, K.: Recognizing syntactic errors in the writing of second language learners. In: Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and the Seventeenth International Conference on Computational Linguistics (ACL/COLING-98), vol. 2, pp. 1198–1204. Montreal, Canada (1998)
39. Smith, N.A., Eisner, J.: Contrastive Estimation: Training log-linear models on unlabeled data. In: Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics, pp. 354–362. Ann Arbor (2005)
40. Smith, N.A., Eisner, J.: Guiding unsupervised grammar induction using contrastive estimation. In: Proceedings of the IJCAI Workshop on Grammatical Inference Applications, pp. 73–82. Edinburgh (2005)
41. Stemberger, J.: Syntactic errors in speech. Journal of Psycholinguistic Research **11**(4), 313–45 (1982)
42. Vogel, C., Cooper, R.: Robust chart parsing with mildly inconsistent feature structures. In: A. Schöter, C. Vogel (eds.) Nonclassical Feature Systems, pp. 197–216. Centre for Cognitive Science, University of Edinburgh, Working Papers in Cognitive Science (1995). Volume 10
43. Weng, F.: Handling syntactic extra-grammaticality. In: Proceedings of the Third International Workshop on Parsing Technologies (IWPT-93), pp. 319–332. Tilburg (the Netherlands), Durbuy (Belgium) (1993)