Parse Tree Fragmentation of Ungrammatical Sentences

Homa B. Hashemi, Rebecca Hwa
Intelligent Systems Program, Computer Science Department
University of Pittsburgh
hashemi@cs.pitt.edu, hwa@cs.pitt.edu

Abstract

Ungrammatical sentences present challenges for statistical parsers because the well-formed trees they produce may not be appropriate for these sentences. We introduce a framework for reviewing the parses of ungrammatical sentences and extracting the coherent parts whose syntactic analyses make sense. We call this task *parse tree fragmentation*. In this paper, we propose a training methodology for fragmenting parse trees without using a task-specific annotated corpus. We also propose some fragmentation strategies and compare their performance on an extrinsic task – fluency judgments in two domains: English-as-a-Second Language (ESL) and machine translation (MT). Experimental results show that the proposed fragmentation strategies are competitive with existing methods for making fluency judgments; they also suggest that the overall framework is a promising way to handle syntactically unusual sentences.

1 Introduction

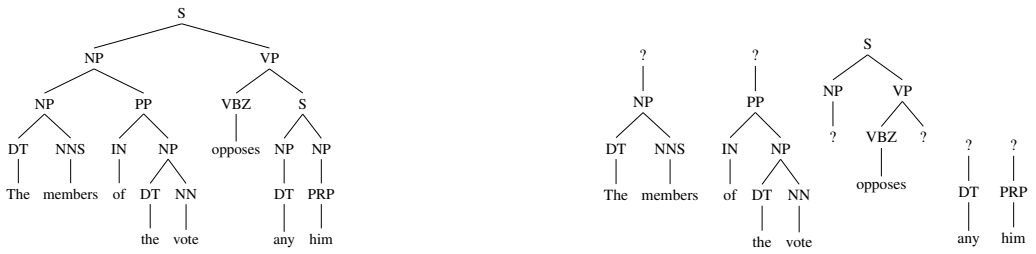
Syntactic parse trees are integral to many Natural Language Processing (NLP) applications. While state-of-the-art statistical parsers perform well on standard (newswire) benchmarks, their analyses for sentences from different domains are less reliable [Gildea, 2001; McClosky *et al.*, 2010; Foster, 2010; Petrov *et al.*, 2010; Foster *et al.*, 2011]. Ungrammatical[1] sentences (or even awkward sentences that are technically grammatical) can be seen as special kinds of out-of-domain sentences; in some cases, it is not even clear whether a complete parse should be given to the sentence. A statistical parser trained on a standard treebank, however, often produces full, syntactically well-formed trees that are not appropriate for the sentences.

Rather than throwing out parses with low-confidence scores entirely or transforming the problematic sentences first (which may not always be possible), we believe that a valuable middle ground is to identify well-formed syntactic structures of those parts of the sentences that do make sense. The resulting reliable structures may still provide some useful information for down-stream NLP applications such as information extraction (IE), machine translation (MT), and automatic evaluation of text (e.g., generated by MT or summarization systems or human second language learners); the omission of the problematic structures also helps to prevent models that learn from syntactic structures from degrading due to incorrect syntactic analysis.

One approach for obtaining these partially completed structures is to use shallow parsing [Abney, 1991; Sha and Pereira, 2003; Sun *et al.*, 2008] to identify recognizable low-level constituents, but this excludes higher-level complex structures. Instead, we propose to review the full parse tree generated by a state-of-the-art parser and identify the parts of it that are plausible interpretations for the phrases they cover. We call these isolated parts of the parse tree *fragments*, and the process of breaking up the tree, *parse tree fragmentation*. Our approach differs from systems that produce partial parses (e.g., the Connexor parser) because we are not using a deterministic grammar building bottom up; we are re-interpreting the parse tree post-hoc. Our task also differs from disfluency detection in spoken utterances, which focuses on removing extra fillers and repeated phrases [Honnibal and Johnson, 2014; Rasooli and Tetreault, 2013; Ferguson *et al.*, 2015]; ungrammatical sentences written by non-native speakers or generated by machines have a wider range of error types, such as missing phrases and incorrect phrasal ordering.

In this paper, we present a framework for extracting parse tree fragments from problematic sentences. We develop a methodology for creating gold standard data for training and evaluating parse tree fragmentation methods. We propose two fragmentation algorithms and compare them in an empirical study. Finally, we perform an extrinsic evaluation to determine the potential utility of tree fragmentation to a downstream application. In particular, we choose the task of sentential fluency judgment, in which a system automatically predicts how "natural" a sentence might sound to a native-speaker human. We consider two domains: MT outputs and the writings of English-as-a-Second Language (ESL) students.

[1] In this context, a sentence is considered ungrammatical if all its words are valid in the language, but it contains grammatical or usage errors [Foster, 2007].



(a) Stanford parse tree

(b) Coherent fragments

Figure 1: (a) An ungrammatical sentence gets a well-formed but inappropriate parse tree. (b) A set of coherent tree fragments that might be extracted from the full parse tree.

2 A Framework for Parse Tree Fragmentation

The goal of parse tree fragmentation is to take a sentence and its tree as input and extract from the tree a set of partial trees that are well-formed and appropriate for the phrases they cover. Before we explore methods for doing so, we need to address some more fundamental problems: What kind of partial trees are considered to be well-formed and appropriate? How do we obtain enough examples of appropriate ways to fragment the trees? How should this task be evaluated?

2.1 Ideal Fragmentation

One factor that dictates how fragmentation should be done is how the fragments will be used in a downstream application. For example, a one-off slight grammar error (e.g., number agreement) probably will not greatly alter a parser output. For the purpose of information extraction, this type of slight mismatches should probably be ignored; for the purpose of training future syntax-based computational models, on the other hand, more aggressive fragmentation may be necessary to filter out unwanted syntactic relationships.

Even assuming a particular downstream application choice (sentential fluency judgment in our case), the ideal fragmentation may not be obvious, especially when the errors interact with each other. Consider the following output from a machine translation system that contains three problem areas (underlined):

The members of the vote opposes any him.

Figure 1a shows the Stanford parser's output for this sentence. The first problem is the subject noun phrase even though the embedded *NP* and *PP* subtrees are well-formed. The second problem is a number disagreement between the subject and the verb. The third problem is an unusual bigram that causes the parser to group *any* and *him* into a sentential clause to serve as the object of the main verb.

Which fragments should be salvaged from this parse tree? Someone who thinks the sentence says: *The members of the voting body oppose any proposal by him* might produce the fragment set shown in Figure 1b. On the other hand, if they think it says: *No parliament members voted against him*, they might have opted to not keep the *PP* intact.

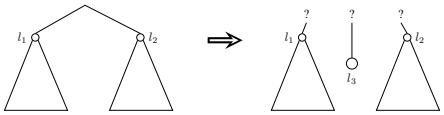
This example illustrates that fragmentation decisions are influenced by the amount of information we glean from the sentence. With only a sentence and an automatically generated tree for it, we may mentally error-correct the sentence in different ways. If we are also given an acceptable paraphrase for the sentence, the fragmentation task becomes more circumscribed because we now know the intended meaning. An example data source of this type is an MT evaluation corpus, which consists of machine-translated sentences and their corresponding human-translated references. Furthermore, if we not only have access to a closely worded paraphrase but also an explanation for each change, the fragmentation decisions are purely deterministic (e.g., whenever a phrase is recommended for deletion, the tree over it is fragmented). An example data source of this type is an ESL learner's corpus, which consists of student sentences and their detailed corrections.

2.2 Developing a Fragmentation Corpus

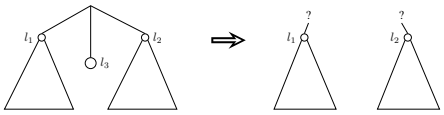
Because the definition of an ideal fragmentation depends on multiple factors (e.g., the intended use and the context in which the original sentences were generated), this task is not well-suited for a large-scale human annotation project. Instead, we propose to develop our fragmentation corpus by leveraging existing data sources previously mentioned (an ESL learner's corpus and an MT evaluation corpus).

Pseudo Gold Fragmentation (PGold) An ESL learner's corpus in which every sentence has been hand corrected by an English teacher is ideal for our purpose. We identified sentences that are marked as containing word-level mistakes: *unnecessary*, *missing* or *replacing* word errors. Given the positions and error types, a fluent sentence can be reconstructed and reliably parsed. The parse tree of the fluent sentence can then be iteratively fragmented according to the error types that occur in the original sentence (see Figure 2). The resulting sets of fragments approximate an explicitly manually created fragmentation corpus; however, since a parser may make mistakes even on a fluent sentence, we call these fragments *pseudo gold*.

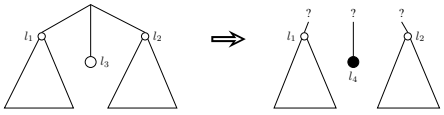
REference Fragmentation (REF) Even if we do not have detailed information about why certain parts of a sentence are problematic, we can construct an almost-as-good fragmentation if we have access to a fluent paraphrase of the original. We call this a *reference* sentence, borrowing the terminology



(a) Unnecessary word (l_3) error



(b) Missing word (l_3) error



(c) Replacing word (l_3) error

Figure 2: Creating pseudo gold fragments. The left-hand side are parse trees of fluent sentences and the right-hand side are their transformation after applying errors.

from the MT community, where it is used to refer to human translations against which MT systems are evaluated. In a language tutoring scenario, the reference would be a teacher's revision of a student's original attempt.

We parse both the original and reference sentences with the same parser. We then find the alignments between two trees by computing their minimum tree edit distance [Pawlik and Augsten, 2011]. Based on the tree alignment, we assign every node of the original tree as either a part of an existing fragment or the start of a new fragment. If the node in a parse tree is aligned, it will be assigned to a fragment along with its aligned parent, children or siblings. If the node is not aligned, it will be considered as a split point between fragments.

While both PGold and REF made use of additional information to create reliable tree fragments, they serve different purposes. PGold tree fragments represent the most linguistically plausible interpretation of the original sentence because we can construct the intended well-formed sentence and obtain the fragments from its corresponding well-formed tree. In contrast, an automatic alignment between an original sentence and a reference sentence may not be as linguistically plausible (e.g., an error could be fixed via a substitution or via an insertion plus a deletion). Therefore, the REF tree fragments are formed from the automatically parsed tree of the original sentence, and they represent an upperbound on what a real fragmentation algorithm could achieve.

2.3 Evaluative Metrics

One way to evaluate an automatic fragmentation method is to compare its resulting fragments against the gold standard fragments, adapting the usual tree-to-tree precision and recall metrics for set-to-set. First, each fragment of the candidate set is mapped to a fragment of the gold standard set with which it has a maximum number of shared edges. (If there are two candidate fragments but only one gold fragment, both candidates would be mapped to the same gold fragment.) Second, precision and recall (and F-score) are calculated as the number of shared edges between all the mapped fragments

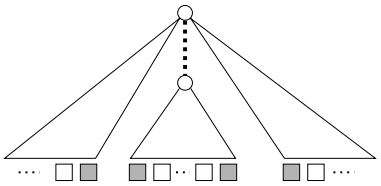


Figure 3: Word N-gram features for the dotted edge. Rectangles are words. Word bigrams associated to the dotted edge are shadowed.

divided by the total number of edges in the candidate and the gold fragment sets respectively. In our work, the PGold set serves as the gold standard. Another way of evaluating whether the fragmentation decisions make sense is to perform an extrinsic evaluation. If fragmentation were helpful, a downstream application should perform better with it than without it.

3 Fragmentation Methods

We propose two automatic methods of fragmentation. In one, we assume the availability of a gold standard training corpus. In the second, we only make use of widely-available NLP resources, such as treebanks.

3.1 Classification-based Parse Tree Fragmentation (CLF)

To automatically extract reliable parse tree fragments from an ungrammatical sentence, a system needs to discriminate between the right and wrong contexts for some parent-child syntactic relationships. We formulate this as a binary classification problem: for each edge in the tree indicates whether the edge should be kept or cut. Using parse trees that were fragmented by the REF method as examples, we train a Gradient Boosting Classifier [Friedman, 2001] that learns to fragment trees in a similar manner as REF. The trained classifier can then make predictions on the branches of unseen parse trees.

Because the number of kept edges is far greater than the cut ones, when constructing the training set, we randomly sample equal numbers of the kept and cut edges. The following features are extracted from each parent-child edge of a parse tree:

- Labels of the parent, child and grandparent nodes.
- Depth and height of the parent and child.
- Word bigrams and trigrams corresponding to the edge (as shown in Figure 3). We use both raw counts and pointwise mutual information of the N-grams. To compute the N-gram counts, we use Agence France Press English Service (AFE) section of English Gigaword [Graff *et al.*, 2003].
- Binary features that determine whether the edge is in a context free grammar rule that appeared more than some threshold in the Penn Treebank. We used these 12 values: $\{100, 500, 1000, 3000, 5000, 8000, 10000, 15000, 20000, 40000, 70000, 90000\}$.

3.2 Treebank-Based Parse Tree Fragmentation (TBF)

For domains that do not have parallel corpora, we would need to back off to widely available resources, such as treebanks. The Treebank-Based Fragmentation method (TBF) decides whether to keep or cut an edge in a parse tree by checking to see whether it belongs to a "common" CFG rule in the Penn Treebank; a CFG rule is considered "common" if its treebank frequency is above a set threshold value (empirically chosen using development data). If the rule is not common but has a common right-hand side, the right-hand side will be kept as a fragment. Otherwise, TBF finds a pattern within the rule with the highest frequency; if it is common, it is kept as a fragment. For example, the rule *NP* $\rightarrow$ *NP PP , VP* is not common, nor is its entire right-hand side. Therefore, TBF finds the most frequent pattern within the rule, which is *NP* $\rightarrow$ *NP PP*. Because this pattern is common, it is kept as a fragment, while other children (the comma and *VP*) are separated into two other fragments.

4 Experimental Setup

4.1 Data

The experiments are conducted over two datasets that contain ungrammatical sentences: writings of English as a second language learners and machine translation outputs. We choose datasets for which the corresponding correct sentences are available (or easily reconstructed).

English as a Second Language corpus (ESL) The First Certificate in English (FCE) dataset [Yannakoudakis *et al.*, 2011] is a learner's corpus that contains ungrammatical sentences and their corresponding error corrections. Given the location and type of the errors, a corrected version of each ungrammatical ESL sentence can be reconstructed. For example, in a sentence "He talk with a friend" the teacher would annotate that "talk" should be replaced by "talks" because it has the wrong number agreement. In most cases, knowing the errors and their corrections makes it possible for us to determine the appropriate fragments. However, some corrections are more complicated, involving phrase-to-phrase replacement due to multiple problems. For example, suppose a teacher recommended replacing "have a talk" with "talked". This edit involves both a semantic shift as well as a tense change. On a more micro-level, should the corrected verb "talked" be aligned with the original noun "talk" (because they are more semantically similar) or the original verb "have" (because they are more syntactically similar)? In this study, we filter out these phrasal replacements, leaving them for future work. From this corpus, we create two datasets for the experiments. First, we randomly select 5000 sentences with at least one error; this dataset is for training the CLF fragmentation method as well as for the intrinsic evaluation of different fragmentation methods. Then, we create a second, non-overlapping dataset for the extrinsic evaluation. It consists of 7000 sentences and is representative of the corpus's error distribution; there are 2895 sentences with no error; 2103 with one error; 1092 with two errors; and 910 with 3+ errors.

Machine Translation corpus (MT) The LIG corpus [Potet *et al.*, 2012] contains 10,881 French-English machine translation outputs and their human post-editions. Unlike the ESL corpus, in the MT corpus, we only have access to the human-edited sentence. We cannot create PGold fragmentation for the MT data because we are not certain about positions or types of the errors. We can only build REF fragments for MT by comparing the parse tree of the bad sentence with that of the good sentence, making splitting point decisions on the parse tree of the bad sentence.

In our experiments on the MT corpus, we use the HTER (Human-targeted Translation Edit Rate) score [Snover *et al.*, 2006][2] as the fluency score of MT outputs. HTER is defined as the minimal rate of edits needed to change the machine translation to its manually post-edited version that ranges between 0 and 1 (0 when no word is edited and 1 when all words are edited). We use TER (default settings)[3] to compute HTER on the LIG corpus. We then build a dataset of 4000 sentences with HTER score more than 0.1 for training CLF and intrinsic evaluation, and a dataset of 6000 sentences with real distribution of HTER scores for extrinsic evaluation. The distribution of sentences in the second dataset is as follows: The HTER score of 2109 sentences are within $[0, 0.1)$; 1099 sentences within $[0.1, 0.2)$; 1195 sentences within $[0.2, 0.3)$; 784 sentences within $[0.3, 0.4)$; and 813 sentences have scores more than 0.4.

4.2 Experimental Tools and Methods

The Stanford parser version 3.2.0 [Klein and Manning, 2003] is used to generate parses for all sentences. In a post-processing step, the non-terminal symbols and terminal words of the tree are normalized; similarly, we have also lemmatized all the words. This increases the number of node alignments for the REF and CLF methods and the number of hits for the TBF methods[4]. We remove all fragments that do not have any terminals, thus all fragments are lexicalized. In addition, in each fragment, if head of a constituent is not among its children, its children will form new fragments.

For all binary classification or regression tasks (which will be discussed in Section 5.2), we run a 10-fold cross validation with the standard Gradient Boosting Classifier or Regressor [Friedman, 2001] in the scikit-learn toolkit [Pedregosa *et al.*, 2011].[5] We tune Gradient Boosting parameters with a 3-fold cross validation on the training data: `learning_rate` over the range $\{0.0001 \ldots 100\}$ by multiples of 10 and `max_depth` over the range $\{1 \ldots 5\}$.

[2] This score is also used in Workshop on Statistical Machine Translation (WMT) for the sentence-level quality estimation task.

[3] http://www.cs.umd.edu/~snover/tercom/

[4] Normalization may allow some low-level grammar errors to pass through. Because our goal is to produce large, interpretable fragments with an eye toward a fluency judgment, normalization helps more than hurts. However, if the application were grammar error correction, alternative preprocessing may be necessary.

[5] We have also tried SVMs with LibLinear toolkit [Fan *et al.*, 2008], but gradient boosting learners obtained the best results.

method	avg. # of fragments	avg. size of fragments	F-score
PGold	6	10.9	-
REF	5.7	13.2	0.86
CLF	7.1	9.3	0.74
TBF	8.9	7.8	0.72
No cut	1	55.8	0.65

Table 1: Similarity of fragmentation methods with PGold fragments over ESL dataset. The *No cut* method serves as a floor baseline and does not break any tree.

5 Evaluation

To measure how well the proposed CLF fragmentation method performs, we have conducted both an intrinsic evaluation and an extrinsic one. For the intrinsic evaluations, we first validate CLF using standard classification measures; we then compare its tree fragments against those produced by other fragmentation methods. For an extrinsic evaluation, we use the fragments for a specific NLP task: sentence-level fluency judgment.

5.1 Intrinsic Evaluation

Evaluation of CLF Performance CLF runs a binary prediction model over parse tree edges, deciding whether to keep an edge or cut it. The ground-truth labels come from the REF fragments. We performed a 10-fold cross validation for the two domains: ESL and MT. Note that while the CLF training data is balanced, the test data is not; thus, a baseline of never cutting any edge would result in a high classification accuracy (84% on ESL and 66% on MT). To take the skewed class distribution into account, we evaluate classifiers with the AUC measure (the area under the receiver operating characteristic curve) [Hanley and McNeil, 1982]. AUC estimates how probable it is that a classifier might give a higher rank to a randomly cut-edge compared to a randomly not-cut-edge. In our experiments, the AUC of CLF on ESL and MT is 0.69 and 0.66 respectively whereas the AUC of the baseline (cutting no edge) is 0.5 for both. The AUC scores suggest that CLF is making reasonable decisions, opting to cut an edge when it is certain.

Evaluation of Tree Fragmentation Methods In this experiment, we evaluate the fragmentation methods by how well their resulting tree fragments match the PGold tree fragments, which are extracted from the ESL learner's corpus. To perform the comparison, we use an adapted version of the usual precision and recall metrics (described in Section 2.3). Table 1 summarizes the comparison of different fragmentation methods in terms of their average number of fragments, average fragment size, and F-score against PGold fragments. We see that while REF fragments are the most similar to PGold, they are far from identical. This highlights the differences that arise from fragmenting a well-formed tree (PGold) and fragmenting the tree of the problematic sentence (REF).

5.2 Extrinsic Evaluation: Fluency Judgment

The previous intrinsic evaluation only tells us how closely an automatic tree fragmentation method might approach the PGold fragments. Since even the PGold fragments are automatically created, we evaluate the potential uses of tree fragments in an external application: sentence-level fluency judgment. An automatic fluency judge can be used to decide whether an MT output needs to be post-processed by a professional translator; it can also be used to help grading student writings.

There have been several previous work on sentence-level fluency judgment. Researchers have found that language model metrics alone are not sufficient, and various syntax-based features have been proposed to be incorporated into the fluency metric [Mutton *et al.*, 2007; Post, 2011; Post and Bergsma, 2013]. However, in order for these features to work well, they ought to be extracted from appropriate parse trees. Given that statistical parsers have difficulties with ungrammatical sentences, mis-interpreted parse trees may degrade the predictive power of the features. We hypothesize that through parse tree fragmentation, major syntactic problems can be identified; thus, tree fragments should be useful for judging sentence fluency.

There are many different ways to set up a fluency judgment task. For example, the desired granularity of the judgment may differ depending on the application. We report two conditions: an easy binary classification and a regression formulation. For the binary classification task, we train a classifier to distinguish between sentences that have virtually no error and those that have many errors. Thus, an ESL sentence is labeled 0 if it has no errors, and it is labeled 1 if it has three or more errors; an MT output is labeled 0 if its HTER score is less than 0.1, and it is labeled 1 if its HTER score is greater than 0.4. Although the setup is a little artificial, this study tells us how well each method performs on the extreme cases. In contrast, the regression task is more challenging because the systems have to make finer distinctions of fluency. For the ESL dataset, the system has to predict the number of errors in each sentence (0, 1, 2, or 3+); for the MT dataset, the HTER score (a real number between 0 and 1). The reported metric for this study is Pearson's r correlation coefficient between the predicted and expected values.[6]

Our feature set: We extract four simple features from the output of each fragmentation method for each sentence: 1) Number of fragments, 2) Average size of fragments, 3) Minimum size of fragments, and 4) Maximum size of fragments.

Contrastive feature sets: We compare the proposed fragmentation approach against several contrastive baselines. In addition to typical language model features, we especially focused on previous work that rely on parse information:

- Sentence length (l).
- Language Modeling (LM). An N-gram precision for $1 \leq N \leq 5$ is computed as a fraction of N-grams appearing in the reference text (we used the Agence France

[6]We have also evaluated the regression task with root mean square error (RMSE) and Kendall's τ. Since the general trend of the results was similar to Pearson's r, we only report Pearson's r.

feature set	ESL Acc.(%)	ESL AUC	ESL r	MT Acc.(%)	MT AUC	MT r
Chance	76.1	0.5		72.2	0.5	
length (l)	77.3	0.75	0.304	72	0.5	0.018
LM	76.7	0.73	0.279	74.4	0.71	0.307
LM + l	80.6	0.84	0.417	74.2	0.71	0.306
C&J	76.3	0.74	0.318	68.3	0.6	0.136
TSG	77.3	0.74	0.285	69.8	0.59	0.105
PGold	100	1	0.928	-	-	-
REF	99.8	1	0.84	94.4	0.99	0.782
CLF	79.9	0.81	0.377	73	0.66	0.205
TBF	77.2	0.74	0.298	71.8	0.51	0.04
TBF + LM	81.1	0.84	0.45	74	0.71	0.304
CLF + LM	**82.2**	**0.86**	**0.462**	74.7	**0.73**	**0.324**

Table 2: Fluency judgment results over two datasets containing ungrammatical sentences using binary classification and regression. Accuracy and AUC measures are reported for binary classification, and Pearson's r is reported for regression.

Press English Service (AFE) section of the English Gigaword Corpus [Graff *et al.*, 2003].

- C&J features (C&J). This set of features is based on the complete set of parse tree reranking features of [Charniak and Johnson, 2005][7] from Stanford parser's output.
- TSG features (TSG). This set of features is based on the tree substitution grammar derivation counts [Post, 2011].[8]

Table 2 summarizes a comparison of different fluency judgment feature sets. The first block reports the baselines. For the ESL domain, the length of a sentence is a good indicator of the fluency of a sentence; longer sentences tend to have more errors than shorter sentences, but sentence length is not as strongly correlated with HTER score in the MT domain.

The second block of feature sets in the table shows that the four features extracted from parse tree fragments are correlated with the fluency quality of sentences. While it is expected that features based on PGold and REF fragments should correlate strongly with fluency, CLF features also correlates with fluency better than C&J and TSG features in both domains. Moreover, they have different model sizes: the CLF feature set consists of 4 simple extracted features from the CLF tree fragments, C&J has more than 60k features and TSG has more than 6k features on the ESL dataset. As a simpler fragmentation heuristic, TBF is not as competitive for fluency judgment, especially for the MT domain. This suggests that CLF has learned useful signals from the REF training examples.

Finally, adding parse tree fragmentation features to language modeling features improves the overall fluency judgment performance. As the results show, the combination of CLF features with LM features ($CLF + LM$) significantly outperforms $LM+l$ (using two-sided paired t-test with $> 95\%$ confidence from the 10 folds).

[7] https://github.com/mjpost/extract-spfeatures

[8] https://github.com/mjpost/post2011judging

6 Related Work

Parsing ungrammatical sentences can be considered as an instance of domain adaptation, in which the goal is to adapt a standard parser to accurately process the ungrammatical text [Foster *et al.*, 2008]. The ungrammatical text might be considered as the target domain that contains the language that is not covered by the parser's grammar. One of the challenges of parser adaptation is the lack of training data for the target domain. Therefore, various approaches have been proposed to automatically label data in the target domain to use as training data. These approaches include self-training [McClosky *et al.*, 2006], parser ensemble [Sagae and Tsujii, 2007; Baucom *et al.*, 2013], selecting source sentences that are most similar to a target domain [McClosky *et al.*, 2010], and building ungrammatical treebank by automatically generating errors to grammatical sentences [Foster, 2007]. The PGold fragmentation in this paper is inspired by [Foster, 2007] in which we iteratively fragment parse trees according to error types. The task of *parse tree fragmentation* can also be considered as an approach for parser adaptation with ungrammatical inputs. Having a treebank of ungrammatical sentences and their parse tree fragments, one might train a new specialized fragmentation parser of ungrammatical sentences.

7 Conclusion

We have introduced parse tree fragmentation as a way to address the mismatch between ungrammatical sentences and statistical parsers that are not trained to handle them. We have devised methods for extracting gold standard tree fragments using evaluative corpora available for other NLP applications. The gold standard corpus enables us to train and evaluate other fragmentation methods. We have proposed two practical fragmentation methods, a classifier-trained method and a deterministic treebank method. Through empirical studies, we have verified that the automatically extracted tree fragments are competitive for an NLP application of fluency detection.

Acknowledgments

This work was supported in part by the National Science Foundation Grant IIS-0745914. We would like to thank the anonymous reviewers and Pitt NLP students for their helpful comments.

References

[Abney, 1991] Steven P Abney. Parsing by chunks. In *Principle-Based Parsing*, 1991.

[Baucom *et al.*, 2013] Eric Baucom, Levi King, and Sandra Kübler. Domain adaptation for parsing. In *RANLP*, 2013.

[Charniak and Johnson, 2005] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*, 2005.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Ferguson *et al.*, 2015] James Ferguson, Greg Durrett, and Dan Klein. Disfluency detection with a Semi-Markov model and prosodic features. In *NAACL*, 2015.

[Foster *et al.*, 2008] Jennifer Foster, Joachim Wagner, and Josef Van Genabith. Adapting a WSJ-trained parser to grammatically noisy text. In *ACL*, 2008.

[Foster *et al.*, 2011] Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, Josef Van Genabith, et al. # hardtoparse: POS tagging and parsing the twitterverse. In *Workshop On Analyzing Microtext (AAAI)*, 2011.

[Foster, 2007] Jennifer Foster. Treebanks gone bad. *International Journal of Document Analysis and Recognition*, 10:129–145, 2007.

[Foster, 2010] Jennifer Foster. "cba to check the spelling" investigating parser performance on discussion forum posts. In *NAACL*, 2010.

[Friedman, 2001] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[Gildea, 2001] Daniel Gildea. Corpus variation and parser performance. In *EMNLP*, 2001.

[Graff *et al.*, 2003] David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English Gigaword. *Linguistic Data Consortium*, 2003.

[Hanley and McNeil, 1982] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[Honnibal and Johnson, 2014] Matthew Honnibal and Mark Johnson. Joint incremental disfluency detection and dependency parsing. *TACL*, 2:131–142, 2014.

[Klein and Manning, 2003] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *ACL*, 2003.

[McClosky *et al.*, 2006] David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *ACL*, 2006.

[McClosky *et al.*, 2010] David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *NAACL*, 2010.

[Mutton *et al.*, 2007] A. Mutton, M. Dras, S. Wan, and R. Dale. GLEU: Automatic evaluation of sentence-level fluency. In *ACL*, 2007.

[Pawlik and Augsten, 2011] Mateusz Pawlik and Nikolaus Augsten. RTED: A robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

[Pedregosa *et al.*, 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Petrov *et al.*, 2010] Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. Uptraining for accurate deterministic question parsing. In *EMNLP*, 2010.

[Post and Bergsma, 2013] Matt Post and Shane Bergsma. Explicit and implicit syntactic features for text classification. In *ACL*, 2013.

[Post, 2011] M. Post. Judging grammaticality with tree substitution grammar derivations. In *ACL (short paper)*, 2011.

[Potet *et al.*, 2012] Marion Potet, Emmanuelle Esperança-Rodier, Laurent Besacier, and Hervé Blanchon. Collection of a large database of French-English SMT output corrections. In *LREC*, pages 4043–4048, 2012.

[Rasooli and Tetreault, 2013] Mohammad Sadegh Rasooli and Joel R Tetreault. Joint parsing and disfluency detection in linear time. In *EMNLP (short paper)*, 2013.

[Sagae and Tsujii, 2007] Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL*, 2007.

[Sha and Pereira, 2003] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *NAACL*, 2003.

[Snover *et al.*, 2006] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Association for Machine Translation in the Americas*, pages 223–231, 2006.

[Sun *et al.*, 2008] Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii. Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *International Conference on Computational Linguistics*, pages 841–848, 2008.

[Yannakoudakis *et al.*, 2011] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading ESOL texts. In *ACL*, 2011.