Jointly Parse and Fragment 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

This paper is about detecting incorrect arcs in a dependency parse for sentences that contain grammar mistakes. Pruning these arcs results in well-formed parse fragments that can still be useful for downstream applications. We propose two automatic methods that jointly parse the ungrammatical sentence and prune the incorrect arcs: a parser retrained on a parallel corpus of ungrammatical sentences with their corrections, and a sequence-to-sequence method. Experimental results show that the proposed strategies are promising for detecting incorrect syntactic dependencies as well as incorrect semantic dependencies.

1 Introduction

Many NLP applications benefit from dependency-based syntactic structures. However, the sentences under analysis may not always be grammatically correct. When a dependency parser nonetheless produces fully connected, syntactically well-formed trees for these sentences, the trees may be inappropriate and lead to errors. In fact, researchers have raised valid questions about the merit of annotating dependency trees for ungrammatical sentences (Ragheb and Dickinson 2012; Cahill 2015). On the other hand, previous work has found that when those dependency arcs directly connected to the erroneous words are ignored, the rest of the parses tend to be well-formed (Hashemi and Hwa 2016a). Thus, for downstream applications that are sensitive to syntactic relationships, it might be preferable to have a set of well-formed parse fragments instead of the entire tree containing errors.

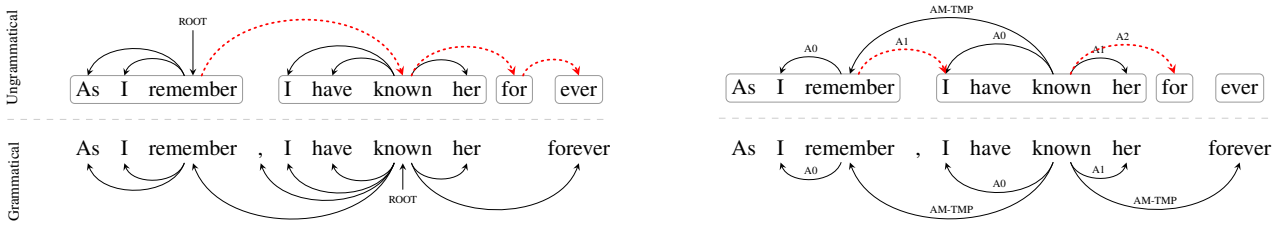
In an earlier work, we have introduced a framework for extracting fragments from constituency parse trees (Hashemi and Hwa 2016b). We proposed a practical fragmentation method using a feature-based classifier and provided an oracle upper bound. However, that study has several limitations. First, there is a significant performance gap between the practical method and the oracle. Perhaps a more sophisticated learning model is necessary. Second, the framework is a pipeline: fragmentation is performed as a post-hoc process on the outputs of off-the-shelf parsers. Perhaps a joint approach that directly builds parse fragments would result in better outputs overall. Third, we have validated the framework on only one downstream application, sentential grammaticality judgment, which does not directly test for the validity of the tree fragments themselves.

In this paper, we address the limitations of the previous work by proposing to jointly parse and fragment ungrammatical sentences to avoid cascading parser errors on these sentences. We introduce two methods to prune implausible dependency arcs of ungrammatical sentences that are caused by grammar mistakes: a parser retraining method and a sequence-to-sequence (seq2seq) labeling method. Both are developed based on a parallel corpus of ungrammatical sentences and their corrections. To better validate our hypothesis that the omission of implausible dependency arcs will directly help downstream applications, we compare the effects of full parses and tree fragments of ungrammatical sentences on the task of semantic role labeling. Through our experiments, we find that both joint methods produce tree fragment sets that are more similar to those produced by the oracle method than the previous pipeline method; moreover, the seq2seq method's pruning decision has a significantly higher accuracy. In terms of downstream applications, we show that dependency arc pruning is helpful for two applications: sentential grammaticality judgment and semantic role labeling.

2 Dependency Arc Pruning

Figure 1a shows an example of an ungrammatical sentence written by an English-as-a-Second Language (ESL) learner and its dependency parse tree, along with the corrected version of the sentence and its parse tree.[1] The ungrammatical sentence has two small mistakes (a missing comma and a phrase replacement error), but the impact of these mistakes is significant on the syntactic parse. Even though the parse tree of the ungrammatical sentence *seems* well-formed, the syntactic structure does not closely resemble the analysis for the corrected sentence: the head of the ungrammatical sentence is changed to "remember" from "known", and the "for ever" phrase is now seen as a preposition relation instead of a time adverb. These errors further impact the semantic interpretation of the sentence. Figure 1b shows the same ungrammatical sentence along with its semantic de-

[1] Dependency trees are produced by SyntaxNet parser (Andor et al. 2016).



(a) Syntactic dependency tree

(b) Semantic dependency graph

Figure 1: Example of syntactic and semantic dependencies of an ungrammatical sentence (top) and its corresponding grammatical sentence (bottom). The red dotted relations show incorrect dependencies, and the word blocks show fragments' boundaries.

pendency graph, as compared to the correct version.[2] Because of the mistakes in the sentence, the semantic graph of the ungrammatical sentence has some extra semantic dependencies: "remember→I" and "known→for". Detecting these incorrect semantic dependencies is crucial for applications that require a high accuracy. For example, some search engines try to display a concise answer to a user's query in addition to retrieving relevant documents; for these systems, it is better to not display any answer at all than to show incorrect answers (since they will still retrieve relevant documents). Therefore, it is important to extract only accurate semantic dependencies.

Our goal is to identify and prune the syntactic dependency arcs of the ungrammatical sentences that are related to the grammar mistakes. The result is a set of *tree fragments* that are linguistically appropriate for the phrases they cover. Since traditionally extracted features from syntactic parse trees play an important role in exploiting semantic dependencies (Punyakanok, Roth, and Yih 2008), we hypothesize that arc pruning would also decrease inaccuracies in semantic role labeling.

In prior work, dependency arc pruning has been explored primarily in the form of vine parsing (Eisner and Smith 2005; Dreyer, Smith, and Smith 2006), where a hard constrain on arc lengths considers only close words as modifiers. Our approach differs from vine parsing in that we do not have any limit on arc lengths; we identify the incorrect arcs with regard to grammar mistakes. We also do not try to correct grammar mistakes (Sakaguchi, Post, and Van Durme 2017), since error detection methods mostly work for ESL error categories and non-ESL mistakes are not easily fixable; we aim to salvage well-formed syntactic structures form ungrammatical sentences in general for downstream applications that use syntactic relationships. Our arc pruning task also differs from joint parsing and disfluency detection in spoken utterances, which focuses on removing repeated phrases and extra fillers (Rasooli and Tetreault 2014; Honnibal and Johnson 2014; Yoshikawa, Shindo, and Matsumoto 2016); ungrammatical sentences have a wider range of error types such as incorrect phrasal ordering and missing phrases.

[2]Semantic dependency graphs are produced by the semantic role labeler of the Mate toolkit (Björkelund, Hafdell, and Nugues 2009).

3 Parse Tree Fragmentation

This work builds on our earlier framework (Hashemi and Hwa 2016b) that tries to extract well-formed tree fragments from ungrammatical sentences. Where appropriate, we follow the same training and evaluative methods. However, while the earlier fragmentation methods worked with constituency parse trees, we want to adapt the framework for the dependency formalism, whose head-modifier representation offers a clearer linguistic interpretation when dealing with ungrammatical sentences and a closer resemblance to semantic relations. Thus, we need to take some necessary steps to facilitate the adaptation. Below, we describe how we construct the gold standard (which also serves as the oracle upper bound) and the pipelined feature-based classifier in a dependency setting.

Pseudo Gold Annotations (Reference) One issue to address in developing tree fragmentation methods is acquiring sufficient training data. Previously, we have proposed a "pseudo gold standard"[3] creation procedure that makes use of a parallel corpus of ungrammatical sentences aligned to their corrected revisions, resulting in a set of *Reference* tree fragments for each sentence pair. We adapt the procedure for dependency trees: given a parallel corpus of ungrammatical sentences and their grammatical versions, we first align the two sentences; then, we parse the grammatical sentence with a state-of-the-art dependency parser, and project the dependency tree to the ungrammatical sentence using the alignments, i.e. for each dependency arc of the grammatical sentence if both the head and the modifier are aligned, we directly project the dependency arc to the ungrammatical sentence. Finally, we apply two restrictive pruning rules (which might be modified depending on a downstream application): 1) prune the dependency arcs to and from the unaligned words, 2) find the immediate right and left words of an unaligned word in the ungrammatical sentence, if there is an arc to or from the right or left words that passes over the unaligned word, prune it. Although these rules are restrictive, they simplify our argument for the use of arc pruning and, at the same time, they still help us to validate the usefulness of arc pruning in our downstream applications.

[3]It uses parse trees of fluent sentences to approximate an explicitly manually created trees; however, since a parser may make mistakes even on a fluent sentence, it is called "pseudo gold".

Post-hoc Arc Pruning (H&H16) Our previous main tree fragmentation method is a binary classifier that determined whether each edge of a constituency parse tree should be pruned based on a set of features that were extracted from the *Reference* tree fragments. We adapt this method for dependency arc pruning so that we can compare to it in an evaluative experiment. Using the *Reference* tree fragments as examples, we train a Gradient Boosting classifier (Friedman 2001) to predict whether to prune each dependency arc according to the following set of features:

- Depth and height of the head/parent and the modifier/child when the dependency tree is traversed in depth-first order.
- Part-of-speech tags of the head, modifier, and the parent of the head word.
- Word bigrams and trigrams corresponding to the arc. Denoting w_h as the head word and w_m as the modifier word, the bigram feature are calculated for the pairs of $w_h w_m$ ($w_m w_h$ if $m < h$), $w_{m-1} w_m$, and $w_m w_{m+1}$. The trigram features are calculated for the triples of $w_{m-1} w_m w_{m+1}$, $w_{m-2} w_{m-1} w_m$, and $w_m w_{m+1} w_{m+2}$. We use both raw counts and point-wise 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).

The tree fragments obtained in this post-hoc manner are referred to as *H&H16*.

4 Joint Arc Pruning Methods

We propose two fully end-to-end data-driven approaches to automatically generate arc pruned dependency tree of an ungrammatical sentence. The methods jointly learn to parse a sentence and prune implausible head-modifier arcs of the parse tree considering the grammatical errors that might exist in the sentence. In the first method, we adapt a parser with ungrammatical inputs by building a treebank of ungrammatical sentences. In the second one, inspired by the recent works in neural network-based sequence-to-sequence learning (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2014), we use a state-of-the-art LSTM-based recurrent neural network to address the problem of arc pruning.

4.1 Parser Retraining Arc Pruning (RetrainedParser)

One obvious approach to jointly parse and detect error-related dependency arcs is by adapting parsers for ungrammatical inputs. Having a treebank of ungrammatical sentences and their pruned arcs, we can train a new specialized dependency parser for ungrammatical sentences that not only gives us a parse for the grammatical part of the sentence but also can prune error-related dependency arcs.

We first create a treebank of ungrammatical sentences using the dependency trees that are pruned by the Reference method. The head of the pruned arcs are set to be the wall symbol (i.e. root as the heads). For example the CoNLL based format of dependency tree in Figure 1a with its pruned arcs is:

```
1     As          IN      3
2     I           PRP     3
3     remember    VB      0
4     I           PRP     6
5     have        VB      6
6     known       VB      0
7     her         PRP     6
8     for         IN      0
9     ever        RB      0
```

Using this new ungrammatical treebank that is created by the Reference method as examples, we can train any statistical state-of-the-art parser that learns to prune dependency arcs in a similar manner as the Reference. The trained parser can then jointly parse and prune error-related arcs on the unseen input sentences, hence the obtained tree fragments in this manner are referred to as *RetrainedParser*.

4.2 Sequence-to-Sequence Arc Pruning (seq2seq)

Many tasks in natural language processing can be cast as finding an optimal mapping from a source sequence to a target sequence including machine translation (Bahdanau, Cho, and Bengio 2014), sentence compression (Filippova et al. 2015), grammar error correction (Schmaltz et al. 2016), and dialogue systems (Serban et al. 2015). Theoretically, Recurrent Neural Networks (RNN) were always a potential tool to be used for learning a complex and highly non-linear seq2seq mapping. However, due to the problem of vanishing and exploding gradient, RNNs were far away from being practical. Recent advancements of deep structure RNNs are based on using Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) units, addressing the gradient vanishing and the gradient exploding problem; therefore RNNs have rapidly become a versatile tool in natural language processing.

We formulate the dependency arc pruning problem as finding an optimal sequence-to-sequence mapping (henceforth referred to as *seq2seq*), in which the source sequence is simply the ungrammatical input sentence and the target sequence is a linearized one-to-one mapping of the associated dependency tree with pruned arcs. In the following, for the sake of completeness, we first briefly describe the idea of sequence-to-sequence learning with deep neural networks. Next, we describe how we represent the arc pruned dependency trees in a linear form as the target sequence of the seq2seq problem.

Seq2Seq Using Deep Neural Nets We follow the dominant approach of training a seq2seq framework, using a conditional language model and a cross-entropy loss function to maximize the conditional likelihood of a successive target word in the target sequence given the input sequence and a history of target words. Following the past practice of the state-of-the-art seq2seq deep neural network models, in our network architecture, we use a stack of LSTM recurrent networks to encode the input sequence (or to be more accurate, a word embedding of the input sequence) into a latent

representation that would be useful in finding the target sequence. Another stack of LSTM recurrent neural networks is used to decode the encoded latent representation of the input sequence to the target output sequence. For the training, in each step, the error signal generated by the cross-entropy loss function will be back-propagated through the network for tuning the weights to minimize the corresponding empirical risk on a batch of data (Sutskever, Vinyals, and Le 2014).

Sequence Representation of an Arc Pruned Tree We treat dependency arc pruning as a seq2seq task by attempting to map from an input sentence to a linear form of arc pruned dependency tree. Using the ungrammatical sentences and their dependency trees that are pruned by the Reference method, we can train a seq2seq model. The seq2seq models require an effective representation for the input and the output to yield good performance (Vinyals, Bengio, and Kudlur 2015). We therefore follow the representation of (Wiseman and Rush 2016) to linearize dependency trees, by inserting arc-standard reduce actions interleaved with the sentence words as the shift action. As an example, we try to map the input sentence to the output sequence:

Input: As I remember I have known her for ever

Output: As I remember @L @L I have known @L @L her @R for ever @RCUT @RCUT @RCUT

We use unlabeled arcs and show the actions with @L as the left-arc action, and @R as the right-arc action. The pruned arced are denoted by @LCUT and @RCUT actions whether it was a left-arc or a right-arc. A trained seq2seq model with this representation would be able to prune error-related arcs of an ungrammatical sentence while parsing the remaining grammatical parts of the sentence.

In order to evaluate the seq2seq method, we then convert back the output of seq2seq which is in the form of interleaved arc-standard actions to a CoNLL format of dependency tree (similar to the example in the RetrainedParser approach, Section 4.1).

5 Experimental Setup

5.1 Data

The experiments are conducted over writings of English-as-a-Second language (ESL) learners dataset that contain ungrammatical sentences and the corrected version of each ungrammatical ESL sentence. To retrain a parser and the seq2seq model, we need a large amount of ungrammatical sentences. Thus, we used three ESL corpora: First Certificate in English (FCE) dataset (Yannakoudakis, Briscoe, and Medlock 2011), National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier, Ng, and Wu 2013), and EF-Cambridge Open Language Database (EFCAMDAT) (Geertzen, Alexopoulou, and Korhonen 2013). From these corpora, we randomly select 606,000 ESL sentences that 566,000 of them have at least one grammatical error. We then randomly separate 30,000 sentences as the development set, and the remaining 576,000 sentences as the training set. To test the models, we obtain the 7000 sentences of the FCE corpus used in our previous work (Hashemi and Hwa 2016b) which do not have overlap with our training and development datasets. The test data contains 2895 sentences with no error; 2103 with one error; 1092 with two errors; and 910 with 3+ errors.

5.2 Experimental Tools and Methods

The pre-trained SyntaxNet POS tagger and parser (Andor et al. 2016)[4] is used to generate dependency parses for all the sentences. To align ungrammatical sentences with their grammatical versions, we use the monolingual word alignment system (Sultan, Bethard, and Sumner 2014) which aligns related words in the two sentences by exploiting the semantic and contextual similarities of the words.

RetrainedParser Settings We create a treebank of our ESL data using the Reference method (as described in §4.1). We then train the SyntaxNet parser (Andor et al. 2016) which is a transition-based neural network parser, and use its globally normalized training with default parameters. We train the parser on the train set and pick the model with the best unlabeled attachment score on the development set.

seq2seq Settings To train the sequence-to-sequence model, we use the OpenNMT[5] (Klein et al. 2017) package, which is a neural machine translation system utilizing the Torch mathematical toolkit. In our implementation of seq2seq RNNs, we used 2-layer LSTMs with 750 hidden units in each layer both for decoding and encoding modules. We trained the network with a batch size of 48 and a maximum sequence length of 62 and 123 for the source and target sequences, respectively. The sequence length is chosen in a way to cover the 5 standard deviations range from the mean of the length of the source and target sequence. The parameters of the model were uniformly initialized in $[-0.1, 0.1]$, and the L2-normalized gradients were constrained to be ≤ 5 to prevent the gradient exploding effect. In the training phase, the learning rate schedule started at 1 and halved the learning rate after each epoch beyond epoch 10, or once the validation set perplexity no longer improved. We trained the network for up to 30 epochs choosing the model with the lowest perplexity on the validation set as the final model.

5.3 Intrinsic Evaluation Metrics

One way to evaluate an automatic arc pruning method is to compare its resulting fragments against the Reference fragments. We use three metrics for this comparison: 1) **Unlabeled Attachment Score (UAS)**: The standard UAS calculates the percentage of words that have the correct head in the dependency tree. It measures the total performance of an automatic method on jointly parse and prune the error-related arcs. 2) **Accuracy of Pruned Arcs**: We also evaluate the accuracy of the automatic methods only on the pruned

[4] github.com/tensorflow/models/tree/master/syntaxnet/syntaxnet/models/parsey_mcparseface

[5] github.com/opennmt/opennmt

arcs. Precision and recall (and F-score) are calculated as the percentage of correct pruned dependency arcs in the resulting parse tree and the Reference tree respectively. 3) **Set-2-Set F-score**: We adapt the set-2-set F-score from our earlier work for dependency trees (Hashemi and Hwa 2016b). We first map each fragment of the candidate set to a fragment of the Reference set with which it has a maximum number of shared arcs; second, precision and recall (and F-score) are calculated as the number of shared arcs between all the mapped fragments divided by the total number of arcs in the candidate and the Reference fragment sets respectively. We report macro-averaged precision, recall and F-score across the test sentences.

6 Evaluation

The experiments aim to measure how well the proposed arc pruning methods perform. We first perform an intrinsic evaluation, comparing all methods on their arc pruning accuracy. We then perform two extrinsic evaluations – sentential grammaticality judgment and semantic role labeling.

6.1 Validating Dependency Arc Pruning

Intrinsic Evaluation Given an ungrammatical sentence, both the RetrainedParser and seq2seq methods produce a dependency parse tree for it with some pruned arcs. Table 1 shows the performance of the produced dependency trees against the Reference trees with the unlabeled attachment score over the 7000 ESL sentences. The UAS suggests that the dependency trees produced by the seq2seq method are more similar to the Reference trees than RetrainedParser, the parser retraining method, and H&H16, the previous pipeline method. Evaluating the accuracy of only the pruned arcs also suggests that the seq2seq method is making reasonable decisions in opting to prune an arc when it is certain.

In the next experiment, we evaluate the arc pruning methods by how well their resulting tree fragments match the Reference tree fragments. Table 2 summarizes the comparison of different fragmentation methods in terms of their average number of fragments, average fragment size, and set-2-set F-score against Reference fragments. We see that H&H16 over-prunes the dependency trees; as a result, it shows less similarity to the Reference. On the other hand, RetrainedParser is cautious in breaking the trees which results in fewer fragments. One reason is that the retrained parser, SyntaxNet, is a transition-based parser which is designed to assign root as the head to the last remaining words in the stack. Even though we train the parser with a large treebank of ungrammatical sentences with multiple words with root as their heads, the parser still tends not to prune arcs. This result suggests that some adaptations may be necessary for the parser; one possible modification is to add a new action to the transition-based dependency parser that marks pruned arcs without removing the modifiers from the stack (because we need the modifiers to obtain the internal syntactic structure of fragments).

The set-2-set F-score similarity of seq2seq to Reference is 0.83, which indicates it has learned useful signals from the Reference method. But, the seq2seq has on average fewer fragments; which shows it prunes less arcs than the Reference method. The results of Table 1 and Table 2 highlights that the seq2seq is conservative on pruning the error-related arcs but when it makes decisions on pruning an arc, it is almost certain.

method	UAS	Accuracy of pruned arcs P	R	F_1
H&H16	61.36	0.35	0.79	0.48
RetrainedParser	63	0.35	0.53	0.42
seq2seq	**82.4**	0.71	0.57	**0.63**

Table 1: Performance of arc pruning methods by comparing their resulting dependency trees against Reference trees.

method	avg. # of fragments	avg. size of fragments	set-2-set P/R/F_1
Reference	3.51	8.60	-
H&H16	7.29	2.40	0.90/0.57/0.67
RetrainedParser	1.8	13.62	0.77/0.82/0.77
seq2seq	2.92	9.36	0.85/0.85/**0.83**

Table 2: Similarity of arc pruning methods with Reference by comparing produced tree fragment sets.

Extrinsic Evaluation: Grammaticality Judgment We also validate the arc pruning utility by replicating our previous sentential grammatical judgment experiment to predict grammaticality of a sentence. We consider both a binarized and an ordinal level of grammaticality for the sentences. In the binarized case, an ESL sentence is labeled 0 if it has no errors, and it is labeled 1 when it has three or more errors. In the ordinal case, the task is to predict the number of errors in each sentence whether it has 0, 1, 2, 3 or more errors. Four simple features are extracted from the fragments: 1) Number of fragments, 2) Average size of fragments, 3) Maximum size of fragments, and 4) Minimum size of fragments. The Gradient Boosting Classifier or Regressor (Friedman 2001) is then used based on 10-fold cross validation on test data. For the binary task, the accuracy of the classifier is reported, and for the regression task the Pearsons r correlation between the predicted and expected values is reported.

Table 3 shows the results of sentence-level grammaticality experiments on the ESL data. The first block shows the baseline feature sets using constituency trees. The Post method (Post 2011) extracts more than 6000 features based on tree substitution grammar (TSG) derivation counts, and the Charniak&Johnson method (Charniak and Johnson 2005) extracts around 60,000 tree reranking features. The second block of the table shows the dependency arc pruning methods with their four features extracted from the resulting fragmented dependency trees. It is expected that features based on the Reference method correlate strongly with the grammaticality of the sentences, because the Reference method prunes dependency arcs with an extra source of information (i.e. grammatical version of the sentence). The seq2seq method significantly outperforms other methods in the binary task (using a two-sided paired t-test with $> 95\%$ confidence from the 10 folds). Although the seq2seq method

feature set	**Binary** Acc.(%)	**Ordinal** r
Post	77.3	0.285
Charniak&Johnson	76.3	0.318
Reference	*100*	*0.879*
H&H16	79.9	**0.377**
RetrainedParser	77.6	0.3
seq2seq	**81.3**	**0.377**

Table 3: Grammaticality judgment results using binary classification and regression. Reference as the upper bound is given in italics, and the best result among automatic methods is given in bold.

makes more accurate pruning decisions, it preforms comparable with H&H16 in the ordinal task. This is because the four extracted features from fragments are very simple; especially since H&H16 produces more fragments, its number-of-fragments feature becomes a good indicator in the ordinal grammatical prediction. As a simpler arc pruning method, RetrainedParser is not as competitive for grammaticality judgment compared to seq2seq and H&H16. But it is still comparable to the other baseline methods. This suggests that RetrainedParser has still learned some useful signals from the Reference training examples.

6.2 Extrinsic Evaluation: Semantic Role Labeling

To further verify arc pruning utility, we apply it in another downstream NLP application which benefits from syntactic parsing: semantic role labeling (SRL). We hypothesize that through dependency arc pruning, major syntactic problems can be identified; thus, tree fragments should be useful to detect *incorrect dependencies* of semantic role labeling.

Creating pseudo gold semantic dependencies Typically, semantic role labelers are evaluated against a gold standard dataset. However, there is no dataset with annotated semantic dependencies for ungrammatical sentences. Instead, we take the automatically produced semantic relations of a grammatical sentence as "gold standard" and compare the SRL output for the corresponding ungrammatical sentence against it.[6] Even if the "gold standard" semantic dependencies are not perfectly correct, they present the norm from which semantic dependencies of ungrammatical sentences diverge: if two sentences have the same meaning, their semantic dependencies for these sentences should be as close as possible. In keeping with this assumption, we create gold semantic dependencies for an ungrammatical sentence by projecting the semantic dependencies of its grammatical sentence to the ungrammatical sentence. We first run the semantic role labeler of the Mate toolkit (Björkelund, Hafdell, and Nugues 2009) over the grammatical sentences; then, find word alignments between ungrammatical and grammatical sentences. Finally, using the alignments, we project directly semantic dependencies of the grammatical sentence to the ungrammatical sentence, i.e. if a word in the ungrammatical sentence is aligned to a word in the grammatical sentence, we directly project the semantic role of the word in the grammatical sentence to the word in the ungrammatical sentence.

[6]Similar idea is used by (Akbik et al. 2015) to construct semantic dependencies for multiple languages.

Detecting incorrect semantic dependencies In order to utilize arc pruning methods for detecting incorrect semantic dependencies of ungrammatical sentences, we introduce a classifier to discriminate between the right and wrong contexts for some semantic dependencies. We formulate this as a binary classification problem: for each semantic dependency generated by an automatic SRL system indicates whether the dependency is correct or incorrect. We extract the following features for each semantic dependency:

- A binary feature that denotes whether the semantic dependency crosses between parse tree fragments. For example, the semantic dependency of "known→for" in Figure 1b crosses two fragments, while "known→her" does not.
- Label of the semantic dependency (e.g. A0, A1, A2 or AM-LOC) (Bonial et al. 2010).

The first two features are unique for each arc pruning method (since their produced tree fragments are different); while the next set of features is shared between the pruning methods which considers only the semantic dependency:

- Depth and height, POS tags, word bigrams and trigrams corresponding to the semantic dependency (similar to the features discussed in Section 3).

Using pseudo gold semantic dependencies as examples, we train a Gradient Boosting classifier that learns to detect incorrect semantic dependencies. We perform the binary classification on a 10-fold cross validation basis on the test data. Because the number of correct semantic dependencies is greater than the incorrect ones, we make a balanced training set by randomly sampling equal numbers of the correct and incorrect dependencies. While we make the train data to be balanced, the test data is not; thus, a baseline of never detecting incorrect dependencies would result in a high classification accuracy (84%). In order 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). The AUC estimates how probable it is that a classifier might give a higher rank to a randomly incorrect dependency compared to a randomly correct one. The AUC of the classifiers with features extracted from H&H16, RetrainedParser and seq2seq are 0.68, 0.67 and 0.70 respectively whereas the AUC of the baseline (detecting all the semantic dependencies as incorrect) is 0.5. The AUC scores suggest that the classifiers are making reasonable decisions to detect incorrect semantic dependencies of ungrammatical sentences. In the remaining of the section, we further evaluate these classifiers in detecting incorrect semantic dependencies with a metric obtained from the standard SRL evaluation schema.

Method	Overall FDR	1 error Verb	1 error Argument	1 error No role
Basic	12.81	12.5	16.5	10.6
Reference	*3.65*	*3.0*	*9.7*	*4.1*
H&H16	7.40	8.1	**15.5**	7.16
RetrainedParser	7.88	9.7	18.1	6.7
seq2seq	**7.32**	**6.8**	16.7	**6.0**

Table 4: False Discovery Rate (FDR) of arc pruning methods in detecting incorrect semantic dependencies. The second column presents FDR on test sentences with one error where the error occurs on a word taking on a verb role, an argument role, or a word with no semantic role. Reference as the upper bound is given in italics, and the best result among automatic arc pruning methods is given in bold.

Evaluation metric We use the standard CoNLL-2009 evaluation script[7] to compare semantic dependencies of an ungrammatical sentence with its pseudo gold semantic dependencies. The *true positives (TP)* is defined as correctly identified semantic dependencies by both automatic system and the gold standard, and *false positives (FP)* is defined as incorrectly identified semantic dependencies by automatic system when there are not semantic dependencies in the gold standard (a.k.a false alarm or *Type I error*). Similarly, a *false negative* case occurs when the automatic system is missing a dependency that exists in the gold standard. In this research, we are less concerned with the false negatives because we do not have any control over adding new semantic dependencies – applying arc pruning methods will only cut semantic dependencies. While the arc pruning methods may cut some correct semantic dependencies, thus introducing false negative cases, that is less problematic than leaving in incorrect semantic dependencies. Detecting incorrect semantic dependencies is crucial for applications that need high accuracy e.g. by building accurate knowledge bases. Therefore, we monitor the number of false positives to evaluate the helpfulness of arc pruning methods when detecting incorrect semantic dependencies. We use a key metric to measure Type I error: *False Discovery Rate* (FDR) (Murphy 2012), defined as $\frac{FP}{FP+TP}$, i.e., as the ratio of incorrect semantic dependencies out of all the identified semantic dependencies by an automatic SRL system. The smaller the FDR value, the better the system preforms in detecting incorrect semantic dependencies.

Results The experiments aim to address the impact of arc pruning methods to detect incorrect semantic dependencies of ungrammatical sentences. The first row of Table 4 is the baseline method named as *Basic*. The Basic method compares the automatically produced semantic dependencies on the ungrammatical sentences with its pseudo gold semantic dependencies; thus, it shows how the automatic SRL system performs on a domain that contains ungrammatical sentences. The first column of the table shows the overall performances of the methods. The overall results demonstrate

[7] ufal.mff.cuni.cz/conll2009-st/eval09.pl

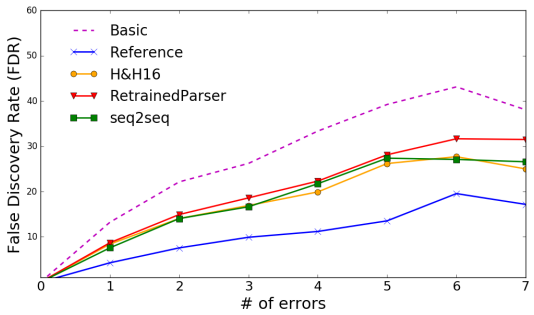


Figure 2: Variation in false discovery rates as the number of errors in the test sentences increases.

that applying arc pruning methods reduces the false discovery rates. This suggests that arc pruning is useful in detecting incorrect semantic dependencies of ungrammatical sentences. The Reference method is outperforming other methods as it uses extra source of information to identify major syntactic problems. When applying automatic models, the seq2seq approach outperforms H&H16 and RetrainedParser. The small differences of FDRs are because the binary classifiers that detect incorrect semantic dependencies use multiple features that are the same for H&H16, RetrainedParser and seq2seq methods; thereby the classifiers are pretty much robust. Also note that, the proposed RetrainedParser and seq2seq methods, not only learn to parse but also learn to prune dependency arcs in a completely automatic regime. Whereas the H&H16 pipelined method only learns to prune dependency arcs using hand-engineered features.

In the next experiment, we examine the impact of semantic role of the error in detecting incorrect semantic dependencies. To remove the impact due to interactivity between multiple errors, we study a subset of sentences that have only one error. An error can be either in a verb role, an argument role, or no semantic role. The second column of Table 4 shows that detecting incorrect semantic roles is more challenging for sentences that have an argument error. It is because the argument errors might not impact the syntactic structure of the sentence, but these errors may change the semantic of the sentence and so make difficulties to detect incorrect semantic dependencies.

We further analyze the results by separating the test sentences by the number of errors each contains. Our objective is to observe the speed with which the rates of false discoveries increases as the sentences become more error-prone. Figure 2 presents false discovery rate plot against the number of errors. We observe that the FDR score is increasing more rapidly for the Basic method than the arc pruning methods especially the Reference method. Therefore, the fact of detecting incorrect semantic dependencies becomes more crucial for the noisier sentences.

7 Conclusions

We have presented two end-to-end data-driven approaches to detect incorrect dependencies of ungrammatical sentences. The methods jointly learn to parse a sentence and prune the

implausible arcs. We devised an evaluation methodology to investigate the utility of arc pruning in the semantic role labeling of ungrammatical sentences. The experimental results validate the utility of the proposed methods, and suggest that the omission of implausible dependency arcs helps to prevent outputs of semantic role labeling to degrade.[8]

Acknowledgments

We would like to thank Allen Schmaltz for helping in running seq2seq experiments and the anonymous reviewers for their helpful comments. This work was supported in part by the National Science Foundation awards #1550635 and #1735752, and the University of Pittsburgh Center for Research Computing through the resources provided. We specifically acknowledge the assistance of Ketan Maheshwari.

References

[Akbik et al. 2015] Akbik, A.; Chiticariu, L.; Danilevsky, M.; Li, Y.; Vaithyanathan, S.; and Zhu, H. 2015. Generating high quality proposition banks for multilingual semantic role labeling. In *ACL*.

[Andor et al. 2016] Andor, D.; Alberti, C.; Weiss, D.; Severyn, A.; Presta, A.; Ganchev, K.; Petrov, S.; and Collins, M. 2016. Globally normalized transition-based neural networks. In *ACL*.

[Bahdanau, Cho, and Bengio 2014] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

[Björkelund, Hafdell, and Nugues 2009] Björkelund, A.; Hafdell, L.; and Nugues, P. 2009. Multilingual semantic role labeling. In *CONLL: Shared Task*.

[Bonial et al. 2010] Bonial, C.; Babko-Malaya, O.; Choi, J. D.; Hwang, J.; and Palmer, M. 2010. Propbank annotation guidelines. *Institute of Cognitive Science, University of Colorad at Boulder*.

[Cahill 2015] Cahill, A. 2015. Parsing learner text: to shoehorn or not to shoehorn. In *LAW IX-Linguistic Annotation Workshop*.

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

[Dahlmeier, Ng, and Wu 2013] Dahlmeier, D.; Ng, H. T.; and Wu, S. M. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *BEA Workshop*.

[Dreyer, Smith, and Smith 2006] Dreyer, M.; Smith, D. A.; and Smith, N. A. 2006. Vine parsing and minimum risk reranking for speed and precision. In *CoNLL*.

[Eisner and Smith 2005] Eisner, J., and Smith, N. A. 2005. Parsing with soft and hard constraints on dependency length. In *Parsing Technology*.

[8]The code is available at: `https://github.com/HHashemi/Dependency_Arc_Pruning`

[Filippova et al. 2015] Filippova, K.; Alfonseca, E.; Colmenares, C. A.; Kaiser, L.; and Vinyals, O. 2015. Sentence compression by deletion with LSTMs. In *EMNLP*.

[Friedman 2001] Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*.

[Geertzen, Alexopoulou, and Korhonen 2013] Geertzen, J.; Alexopoulou, T.; and Korhonen, A. 2013. Automatic linguistic annotation of large scale L2 databases: The EF-cambridge open language database (EFCAMDAT). In *Second Language Research Forum*.

[Graff et al. 2003] Graff, D.; Kong, J.; Chen, K.; and Maeda, K. 2003. English Gigaword. *Linguistic Data Consortium*.

[Hanley and McNeil 1982] Hanley, J. A., and McNeil, B. J. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*.

[Hashemi and Hwa 2016a] Hashemi, H. B., and Hwa, R. 2016a. An evaluation of parser robustness for ungrammatical sentences. In *EMNLP*.

[Hashemi and Hwa 2016b] Hashemi, H. B., and Hwa, R. 2016b. Parse tree fragmentation of ungrammatical sentences. In *IJCAI*.

[Hochreiter and Schmidhuber 1997] Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

[Honnibal and Johnson 2014] Honnibal, M., and Johnson, M. 2014. Joint incremental disfluency detection and dependency parsing. *TACL*.

[Klein et al. 2017] Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. 2017. Opennmt: Open-source toolkit for neural machine translation. In *ACL*.

[Murphy 2012] Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

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

[Punyakanok, Roth, and Yih 2008] Punyakanok, V.; Roth, D.; and Yih, W. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*.

[Ragheb and Dickinson 2012] Ragheb, M., and Dickinson, M. 2012. Defining syntax for learner language annotation. In *COLING*.

[Rasooli and Tetreault 2014] Rasooli, M. S., and Tetreault, J. R. 2014. Non-monotonic parsing of fluent umm I mean disfluent sentences. In *EACL*.

[Sakaguchi, Post, and Van Durme 2017] Sakaguchi, K.; Post, M.; and Van Durme, B. 2017. Error-repair dependency parsing for ungrammatical texts. In *ACL*.

[Schmaltz et al. 2016] Schmaltz, A.; Kim, Y.; Rush, A. M.; and Shieber, S. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. In *BEA Workshop*.

[Serban et al. 2015] Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2015. Building end-to-end dia-

logue systems using generative hierarchical neural network models. *arXiv:1507.04808*.

[Sultan, Bethard, and Sumner 2014] Sultan, M. A.; Bethard, S.; and Sumner, T. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *TACL*.

[Sutskever, Vinyals, and Le 2014] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[Vinyals, Bengio, and Kudlur 2015] Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order matters: Sequence to sequence for sets. *arXiv:1511.06391*.

[Wiseman and Rush 2016] Wiseman, S., and Rush, A. M. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

[Yannakoudakis, Briscoe, and Medlock 2011] Yannakoudakis, H.; Briscoe, T.; and Medlock, B. 2011. A new dataset and method for automatically grading ESOL texts. In *ACL*.

[Yoshikawa, Shindo, and Matsumoto 2016] Yoshikawa, M.; Shindo, H.; and Matsumoto, Y. 2016. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In *EMNLP*.