Parsing Paraphrases with Joint Inference (extended version)

Do Kook Choe*
Brown University
Providence, RI
dc65@cs.brown.edu

David McClosky
IBM Research
Yorktown Heights, NY
dmcclosky@us.ibm.com

Abstract

Treebanks are key resources for developing accurate statistical parsers. However, building treebanks is expensive and time-consuming for humans. For domains requiring deep subject matter expertise such as law and medicine, treebanking is even more difficult. To reduce annotation costs for these domains, we develop methods to improve cross-domain parsing inference using paraphrases. Paraphrases are easier to obtain than full syntactic analyses as they do not require deep linguistic knowledge, only linguistic fluency. A sentence and its paraphrase may have similar syntactic structures, allowing their parses to mutually inform each other. We present several methods to incorporate paraphrase information by jointly parsing a sentence with its paraphrase. These methods are applied to state-of-the-art constituency and dependency parsers and provide significant improvements across multiple domains.

Extended version: This extends the ACL (2015) paper by the same authors. It includes additional analysis of the paraphrase dataset and paraphrase examples in the appendices.

1 Introduction

Parsing is the task of reconstructing the syntactic structure from surface text. Many natural language processing tasks use parse trees as a basis for deeper analysis.

The most effective sources of supervision for training statistical parsers are treebanks. Unfortunately, treebanks are expensive, time-consuming to create, and not available for most domains. Compounding the problem, the accuracy of statistical parsers degrades as the domain shifts away from the supervised training corpora (Gildea, 2001; Bacchiani et al., 2006; McClosky et al., 2006b; Surdeanu et al., 2008). Furthermore, for domains requiring subject matter experts, e.g., law and medicine, it may not be feasible to produce large scale treebanks since subject matter experts generally don't have the necessary linguistic background. It is natural to look for resources that are more easily obtained. In this work, we explore using paraphrases. Unlike parse trees, paraphrases can be produced quickly by humans and don't require extensive linguistic training. While paraphrases are not parse trees, a sentence and its paraphrase may have similar syntactic structures for portions where they can be aligned.

We can improve parsers by jointly parsing a sentence with its paraphrase and encouraging certain types of overlaps in their syntactic structures. As a simple example, consider replacing an unknown word in a sentence with a synonym found in the training data. This may help disambiguate the sentence without changing its parse tree. More disruptive forms of paraphrasing (e.g., topicalization) can also be handled by not requiring strict agreement between the parses.

In this paper, we use paraphrases to improve parsing inference within and across domains. We develop methods using dual-decomposition (where the parses of both sentences from a dependency parser are encouraged to agree, Section 3.2) and pair-finding (which can be applied to any n-best parser, Section 3.3). Some paraphrases significantly disrupt syntactic structure. To counter this, we examine relaxing agreement constraints and building classifiers to predict when joint parsing won't be beneficial (Section 3.4). We show that paraphrases can be exploited to improve cross-domain parser inference for two state-of-the-art parsers, especially on domains where they perform

*Work performed during an IBM internship.

poorly.

2 Related Work

Many constituency parsers can parse English newswire text with high accuracy (Collins, 2000; Charniak and Johnson, 2005; Petrov et al., 2006; Socher et al., 2013; Coppola and Steedman, 2013). Likewise, dependency parsers have rapidly improved their accuracy on a variety of languages (Eisner, 1996; McDonald et al., 2005; Nivre et al., 2007; Koo and Collins, 2010; Zhang and McDonald, 2014; Lei et al., 2014). There are many approaches tackling the problem of improving parsing accuracy both within and across domains, including self-training/uptraining (McClosky et al., 2006b; Petrov et al., 2010), reranking (Collins, 2000; McClosky et al., 2006b), incorporating word clusters (Koo et al., 2008), model combination (Petrov, 2010), automatically weighting training data (McClosky et al., 2010), and using n-gram counts from large corpora (Bansal and Klein, 2011). Using paraphrases falls into the semi-supervised category. As we show later, incorporating paraphrases provides complementary benefits to self-training.

2.1 Paraphrases

While paraphrases are difficult to define rigorously (Bhagat and Hovy, 2013), we only require a loose definition in this work: a pair of phrases that mean approximately the same thing. Paraphrases can be constructed in various ways: replacing words with synonyms, reordering clauses, adding relative clauses, using negation and antonyms, etc. Table 1 lists some example paraphrases.

There are a variety of paraphrase resources produced by humans (Dolan and Brockett, 2005) and automatic methods (Ganitkevitch et al., 2013). Recent works have shown that reliable paraphrases can be crowdsourced at low cost (Negri et al., 2012; Burrows et al., 2013; Tschirsich and Hintz, 2013). Paraphrases have been shown to help summarization (Cohn and Lapata, 2013), question answering (Duboue and Chu-Carroll, 2006; Fader et al., 2013), machine translation (Callison-Burch et al., 2006), and semantic parsing (Berant and Liang, 2014). Paraphrases have been applied to syntactic tasks, such as prepositional phrase attachment and noun compounding, where the corpus frequencies of different syntactic constructions (approximated by web searches) are used to help disambiguate (Nakov and Hearst, 2005). One method for transforming constructions is to use paraphrase templates.

How did Bob Marley die? What killed Bob Marley?
How fast does a cheetah run? What is a cheetah's top speed?
He came home unexpectedly. He wasn't expected to arrive home like that.
They were far off and looked tiny. From so far away, they looked tiny.
He turned and bent over the body of the Indian. Turning, he bent over the Indian's body.
No need to dramatize. There is no need to dramatize.

Table 1: Example paraphrases from our dataset.

2.2 Bilingual Parsing

The closest task to ours is bilingual parsing where sentences and their translations are parsed simultaneously (Burkett et al., 2010). While our methods differ from those used in bilingual parsing, the general ideas are the same.[1] Translating and paraphrasing are related transformations since both approximately preserve meaning. While syntax is only partially preserved across these transformations, the overlapping portions can be leveraged with joint inference to mutually disambiguate. Existing bilingual parsing methods typically require parallel treebanks for training and parallel text at runtime while our methods only require parallel text at runtime. Since we do not have a parallel paraphrase treebank for training, we cannot directly compare to these methods.

3 Jointly Parsing Paraphrases

With a small number of exceptions, parsers typically assume that the parse of each sentence is independent. There are good reasons for this independence assumption: it simplifies parsing inference and oftentimes it is not obvious how to relate multiple sentences (though see Rush et al. (2012) for one approach). In this section, we present two methods to jointly parse paraphrases without complicating inference steps. Before going into details, we give a high level picture of how jointly parsing paraphrases can help in Figure 1. With

[1] Applying our methods to bilingual parsing is left as future work.

the baseline parser, the parse tree of the target sentence is incorrect but its paraphrase (parsed by the same parser) is parsed correctly. We use rough alignments to map words across sentence pairs. Note the similar syntactic relations when they are projected across the aligned words.

Our goal is to encourage an appropriate level of agreement between the two parses across alignments. We start by designing "hard" methods which require complete agreement between the parses. However, since parsers are imperfect and alignments approximate, we also develop "soft" methods which allow for disagreements. Additionally, we make procedures to decide whether to use the original (non-joint) parse or the new joint parse for each sentence since joint parses may be worse in cases where the sentences are too different and alignment fails.

3.1 Objective

In a typical parsing setting, given a sentence (x) and its paraphrase (y), parsers find $a^*(x)$ and $b^*(y)$ that satisfy the following equation:[2]

$$\begin{aligned} a^*, b^* &= \underset{a \in T(x), b \in T(y)}{\operatorname{argmax}} \; f(a) + f(b) \\ &= \underset{a \in T(x)}{\operatorname{argmax}} f(a) + \underset{b \in T(y)}{\operatorname{argmax}} f(b) \end{aligned} \tag{1}$$

where f is a parse-scoring function and T returns all possible trees for a sentence. f can take many forms, e.g., summing the scores of arcs (Eisner, 1996; McDonald et al., 2005) or multiplying probabilities together (Charniak and Johnson, 2005). The argmax over a and b of equation (1) is separable; parsers make two sentence-level decisions. For joint parsing, we modify the objective so that parsers make one global decision:

$$a^*, b^* = \underset{\substack{a \in T(x), b \in T(y) \\ : \, c(a,b)=0}}{\operatorname{argmax}} \; f(a) + f(b) \tag{2}$$

where c (defined below) measures the syntactic similarity between the two trees. The smaller $c(a, b)$ is, the more similar a and b are. Intuitively, joint parsers must retrieve the most similar pair of trees with the highest sum of scores.

3.1.1 Constraints

The constraint function, c, ties two trees together using alignments as a proxy for semantic information. An alignment is a pair of words from sentences x and y that approximately mean the same thing. For example, in Figure 1, (help^x, help^y) is one alignment and (pestilence^x, disease^y) is another. To simplify joint parsing, we assume the aligned words play the same syntactic roles (which is obviously not always true and should be revisited in future work). c measures the syntactic similarity by computing how many pairs of alignments have different syntactic head relations. For the two trees in Figure 1, we see two different relations: (help $\xrightarrow{x}$ dying, help $\not\xrightarrow{y}$ dying) and (natives $\not\xrightarrow{x}$ dying, natives $\xrightarrow{y}$ dying). The rest have the same relation so $c(a, b) = 2$. As we'll show in Section 5, the constraints defined above are too restrictive because of this strong assumption. To alleviate the problem, we present ways of appropriately changing constraints later. We now turn to the first method of incorporating constraints into joint parsing.

[2] When it is clear from context, we omit x and y to simplify notation.

Set $u^0(i, j) = 0$ for all $i, j \in E$
for *k = 1 to K* **do**
  $a^k = \underset{a \in T(x)}{\operatorname{argmax}} \left(f(a) + \sum_{i,j \in E} u^k(i, j) a(i, j) \right)$
  $b^k = \underset{b \in T(y)}{\operatorname{argmax}} \left(f(b) - \sum_{i,j \in E} u^k(i, j) b(i, j) \right)$
  v, u^{k+1} = UPDATE$(u^k, \delta^k, a^k, b^k)$
  if $v = 0$ **then return** a^k, b^k
return a^K, b^K

function UPDATE (u, δ, a, b)
  $v = 0, u'(i, j) = 0$ for all $i, j \in E$
  for $i, j \in E$ **do**
    $u'(i, j) = u(i, j) - \delta(a(i, j) - b(i, j))$
    if $a(i, j) \neq b(i, j)$ **then** $v = v + 1$
  return v, u'

Algorithm 1: Dual decomposition for jointly parsing paraphrases pseudocode. E is the set of all possible edges between any pair of aligned words. Given ℓ aligned word pairs, $E = \{1, \ldots, \ell\} \times \{1, \ldots, \ell\}$. $a(i, j)$ is one if the ith aligned word is the head of jth aligned word, zero otherwise. $u(i, j)$ is the dual value of an edge from the ith aligned word to the jth aligned word. δ^k is the step size at kth iteration.

3.2 Constraints via Dual Decomposition

Dual decomposition (Rush and Collins, 2012) is well-suited for finding the MAP assignment to equation (2). When the parse-scoring function f

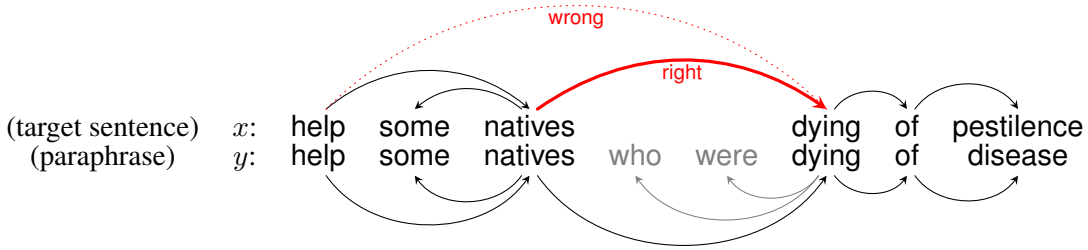

Figure 1: An illustration of joint parsing a sentence with its paraphrase. Unaligned words are gray. Joint parsing encourages structural similarity and allows the parser to correct the incorrect arc.

includes an arc-factored component as in McDonald et al. (2005), it is straightforward to incorporate constraints as shown in Algorithm 1. Essentially, dual decomposition penalizes relations that are different in two trees by adding/subtracting dual values to/from arc scores. When dual decomposition is applied in Figure 1, the arc score of (help $\xrightarrow{x}$ dying) decreases and the score for (natives $\xrightarrow{x}$ dying) increases in the second iteration, which eventually leads the algorithm to favor the latter.

We relax the constraints by employing soft dual decomposition (Anzaroot et al., 2014) and replacing UPDATE in Algorithm 1 with S-UPDATE from Algorithm 2. The problem with the original constraints is they force every pair of alignments to have the same relation even when some aligned words certainly play different syntactic roles. The introduced slack variable lets some alignments have different relations when parsers prefer them. Penalties bounded by the slack tend to help fix incorrect ones and not change correct parses. In this work, we use a single slack variable but it's possible to have a different slack variable for each type of dependency relation.[3]

3.3 Constraints via Pair-finding

One shortcoming of the dual decomposition approach is that it only applies to parse-scoring functions with an arc-factored component. We introduce another method for estimating equation (2) that applies to all n-best parsers.

Given the n-best parses of x and the m-best parses of y, Algorithm 3 scans through $n \times m$ pairs of trees and chooses the pair that satisfies equation (2). If it finds one pair with $c(a, b) = 0$, then it has found the answer to the equation. Otherwise, it chooses the pair with the smallest $c(a, b)$, breaking ties using the scores of the parses ($f(a) + f(b)$). This algorithm is well suited for finding solutions to the equation but the solutions are not necessarily good trees due to overly hard constraints.

The algorithm often finds bad trees far down the n-best list because it is mainly interested in retrieving pairs of trees that satisfy all constraints. Parsers find such pairs with low scores if they are allowed to search through unrestricted space. To mitigate the problem, we shrink the search space by limiting n. Reducing the search space relies on the fact that higher ranking trees are more likely to be correct than the lower ranking ones. Note that we decrease n because we are interested in recovering the tree of the target sentence, x. m should also be decreased to improve the parse of its paraphrase, y.

function S-UPDATE (u, δ, a, b, s)
  $v = 0, u'(i, j) = 0$ for all $i, j \in E$
  for $i, j \in E$ **do**
    $t = \max(u(i, j) - \delta(a(i, j) - b(i, j)), 0)$
    $u'(i, j) = \min(t, s)$
    if $u'(i, j) \neq 0, u'(i, j) \neq s$ **then**
      $v = v + 1$
  return v, u'

Algorithm 2: The new UPDATE function of soft dual decomposition for joint parsing. It projects all dual values between 0 and $s \geq 0$. s is a slack variable that allows the algorithm to avoid satisfying some constraints.

3.4 Logistic Regression

One caveat of the previous two proposed methods is that they do not know whether the original or

[3] We did pilot experiments with multiple slack variables. Since they showed only small improvements and were harder to tune, we stuck with a single slack variable for remaining experiments.

function PAIR-FINDING $(a_{1:n}, b_{1:m})$
  Set $a, b = null, min = \infty, max = -\infty$
  for *i = 1 to* n **do**
    for *j = 1 to* m **do**
      $v = \text{C}\ (a_i, b_j)$
      $sum = f(a_i) + f(b_j)$
      if $v < min$ **then**
        $a = a_i, b = b_j$
        $min = v, max = sum$
      else if $v = min, sum > max$ **then**
        $a = a_i, b = b_j$
        $max = sum$
  return a, b

function C (a, b)
  $v = 0$
  for $i, j \in E$ **do**
    if $a(i, j) \neq b(i, j)$ **then** $v = v + 1$
  return v

Algorithm 3: The pair-finding scheme with a constraint function, c. $a_{1:n}$ are the n-best trees of x and $b_{1:m}$ are the m-best of y.

joint parse of x is more accurate. Sometimes they increase agreement between the parses at the cost of accuracy. To remedy this problem, we use a classifier (specifically logistic regression) to determine whether a modified tree should be used. The classifier can learn the error patterns produced by each method.

3.4.1 Features

Classifier features use many sources of information: the target sentence x and its paraphrase y, the original and new parses of x (a^0 and a), and the alignments between x and y.

Crossing Edges How many arcs cross when alignments are drawn between paraphrases on a plane divided by the length of x. It roughly measures how many reorderings are needed to change x to y.

Non-projective Edges Whether there are more non-projective arcs in new parse (a) than the original (a^0).

Sentence Lengths Whether the length of x is smaller than that of y. This feature exists because baseline parsers tend to perform better on shorter sentences.

REL	REL + REL$_p$
REL + REL$_p$ + REL$_{gp}$	REL + REL$_{gp}$
CP	CP + CP$_p$
CP + CP$_p$ + CP$_{gp}$	CP + CP$_{gp}$
REL + CP	REL + CP + CP$_p$
REL + CP$_p$ + REL$_{gp}$	

Table 2: Feature templates: REL is the dependency relation between the word and its parent. CP is the coarse part-of-speech tag (first two letters) of a word. p and gp select the parent and grandparent of the word respectively.

Word Overlaps The number of words in common between x and y normalized by the length of x.

Parse Structure Templates The feature generator goes through every word in $\{a^0, a\}$ and sets the appropriate boolean features from Table 2. Features are prefixed by whether they come from a^0 or a.

4 Data and Programs

This section describes our paraphrase dataset, parsers, and other tools used in experiments.

4.1 Paraphrase Dataset

To evaluate the efficacy of the proposed methods of jointly parsing paraphrases, we built a corpus of paraphrases where one sentence in a pair of paraphrases has a gold tree.[4] We randomly sampled 4,000 sentences[5] from four gold treebanks: Brown, British National Corpus (BNC), QuestionBank[6] (QB) and Wall Street Journal (section 24) (Francis and Kučera, 1989; Foster and van Genabith, 2008; Judge et al., 2006; Marcus et al., 1993). A linguist provided a paraphrase for each sampled sentence according to these instructions:

> The paraphrases should more or less convey the same information as the original sentence. That is, the two sentences should logically entail each other. The paraphrases should generally use most of the same words (but not necessarily in the same order). Active/passive transforms, changing words with synonyms,

[4] The dataset is available upon request.

[5] We use sentences with 6 to 25 tokens to keep the paraphrasing task in the nontrivial to easy range.

[6] With Stanford's updates: `http://nlp.stanford.edu/data/QuestionBank-Stanford.shtml`

> and rephrasings of the same idea are all examples of transformations that paraphrases can use (others can be used too). They can be as simple as just changing a single word in some cases (though, ideally, a variety of paraphrasing techniques would be used).

We also provided 10 pairs of sentences as examples. We evaluate our methods only on the sampled sentences from the gold corpora because the new paraphrases do not include syntactic trees. The data was divided into development and testing sets such that development and testing share the same distribution over the four corpora. Paraphrases were tokenized by the BLLIP tokenizer. See Table 3 for statistics of the dataset.[7]

4.2 Meteor Word Aligner

We use Meteor, a monolingual word aligner developed by Denkowski and Lavie (2014), to find alignments between paraphrases. It uses the exact matches, stems, synonyms, and paraphrases[8] to form these alignments. Because it uses paraphrases, it sometimes aligns multiple words from sentence x to one or more words from sentence y or vice versa. We ignore these multiword alignments because our methods currently only handle single word alignments. In pilot experiments, we also tried using a simple aligner which required exact word matches. Joint parsing with simpler alignments improved parsing accuracy but not as much as Meteor.[9] Thus, all results in Section 5 use Meteor for word alignment. On average across the four corpora, 73% of the tokens are aligned.

4.3 Parsers

We use a dependency and constituency parser for our experiments: RBG and BLLIP. RBG parser (Lei et al., 2014) is a state-of-the-art dependency parser.[10] It is a third-order discriminative dependency parser with low-rank tensors as part of its features. BLLIP (Charniak and Johnson, 2005) is a state-of-the-art constituency parser, which is composed of a generative parser and a discriminative reranker.[11]

To train RBG and BLLIP, we used the standard WSJ training set (sections 2–21, about 40,000 sentences).[12] We also used the self-trained BLLIP parsing model which is trained on an additional two million Gigaword parses generated by the BLLIP parser (McClosky et al., 2006a).

4.4 Logistic Regression

We use the logistic regression implementation from Scikit-learn[13] with hand-crafted features from Section 3.4.1. The classifier decides to whether to keep the parse trees from the joint method. When it decides to disregard them, it returns the parse from the baseline parser. We train a separate classifier for each joint method.

5 Experiments

We ran all tuning and model design experiments on the development set. For the final evaluation, we tuned parameters on the development set and evaluate them on the test set. Constituency trees were converted to basic non-collapsed dependency trees using Stanford Dependencies (De Marneffe et al., 2006).[14] We report unlabeled attachment scores (UAS) for all experiments and labeled attachment scores (LAS) as well in final evaluation, ignoring punctuation. Averages are micro-averages across all sentences.

5.1 Dual Decomposition

Since BLLIP is not arc-factored, these experiments only use RBG. Several parameters need to be fixed beforehand: the slack constant (s), the learning rate (δ), and the maximum number of iterations (K). We set $\delta^0 = 0.1$ and $\delta^k = \frac{\delta^0}{2^t}$ where t is the number of times the dual score has increased (Rush et al., 2010). We choose $K = 20$. These numbers were chosen from pilot studies. The slack variable ($s = 0.5$) was tuned with a grid search on values between 0.1 and 1.5 with interval 0.1. We chose a value that generalizes well

[7] The distribution over four corpora is skewed because each corpus has a different number of sentences within length constraints. Samples are collected uniformly over all sentences that satisfy the length criterion.

[8] Here paraphrase means a single/multiword phrase that is semantically similar to another single/multiword.

[9] The pilot was conducted on fewer than 700 sentence pairs before all paraphrases were created. We give Meteor tokenized paraphrases with capitalization. Maximizing accuracy rather than coverage worked better in pilot experiments.

[10] `http://github.com/taolei87/RBGParser`, 'master' version from June 24th, 2014.

[11] `http://github.com/BLLIP/bllip-parser`

[12] RBG parser requires predicted POS tags. We used the Stanford tagger (Toutanova et al., 2003) to tag WSJ and paraphrase datasets. Training data was tagged using 20-fold cross-validation and the paraphrases were tagged by a tagger trained on all of WSJ training.

[13] `http://scikit-learn.org`

[14] Version 1.3.5, previously numbered as version 2.0.5

	Development					Test				
	BNC	Brown	QB	WSJ	Total	BNC	Brown	QB	WSJ	Total
Sentences	247	558	843	352	2,000	247	558	844	351	2,000
Tokens	4,297	7,937	8,391	5,924	26,549	4,120	8,025	8,253	5,990	26,388
Tokens$^{\|}$	4,372	8,088	8,438	6,122	27,020	4,272	8,281	8,189	6,232	26,974
Word types	1,727	2,239	2,261	1,955	6,161	1,710	2,337	2,320	1,970	6,234
Word types$^{\|}$	1,676	2,241	2,261	1,930	6,017	1,675	2,335	2,248	1,969	6,094
OOV	11.2	5.1	5.4	2.4	5.6	11.5	5.1	5.8	2.2	5.7
OOV$^{\|}$	8.6	4.7	5.4	2.6	5.1	9.3	4.8	6.0	2.4	5.3
Tokens/sent.	17.4	14.2	10.0	16.8	13.3	16.7	14.4	7.8	17.1	13.2
Avg. aligned	13.1	10.5	6.9	13.0	9.7	12.6	10.7	6.7	13.0	9.7

Table 3: Statistics for the four corpora of the paraphrase dataset. Most statistics are counted from sentences with gold trees, including punctuation. $\|$ indicates the statistic is from the paraphrased sentences. "Avg. aligned" is the average number of aligned tokens from the original sentences using Meteor. OOV is the percentage of tokens not seen in the WSJ training.

	Avg	BNC	Brown	QB	WSJ
RBG	86.4	89.2	**90.9**	75.8	93.7
+ Dual	84.7	87.5	87.8	76.0	91.0
+ S-Dual	**86.8**	**89.8**	**90.9**	**76.5**	**94.0**

Table 4: Comparison of hard and soft dual decomposition for joint parsing (development section, UAS).

n	Avg	BNC	Brown	QB	WSJ
1	89.5	91.1	91.6	83.3	**94.2**
2	**90.0**	91.4	**92.3**	84.1	94.1
3	89.8	91.5	92.0	**84.2**	93.9
5	89.2	**91.9**	91.4	83.0	93.2
10	87.9	90.5	90.3	81.4	92.2
50	86.3	90.2	88.7	78.6	91.1

Table 5: UAS of joint parsing using the pair-finding scheme with various n values on the development portion. $n = 1$ is the baseline BLLIP parser and $n > 1$ is BLLIP with pair-finding.

across four corpora as opposed to a value that does very well on a single corpus. As shown in Table 4, joint parsing with hard dual decomposition performs worse than independent parsing (RBG). This is expected because hard dual decomposition forces every pair of alignments to form the same relation even when they should not. With relaxed constraints (S-Dual), joint parsing performs significantly better than independent parsing. Soft dual decomposition improves across all domains except for Brown (where it ties).

5.2 Pair-finding

These experiments use the 50-best trees from BLLIP parser. When converting to dependencies, some constituency trees map to the same dependency tree. In this case, trees with lower rankings are dropped. Like joint parsing with hard dual decomposition, joint parsing with unrestricted pair-finding ($n = 50$) allows significantly worse parses to be selected (Table 5). With small n values, pair-finding improves over the baseline BLLIP parser.[15] Experiments with self-trained BLLIP exhibit similar results so we use $n = 2$ for all other experiments. Interestingly, each corpus has a different optimal value for n which suggests we might improve accuracy further if we know the domain of each sentence.

5.3 Logistic Regression

The classifier is trained on sentences where parse scores (UAS) of the proposed methods are higher or lower than those of the baselines[16] from the development set using leave-one-out cross-validation. We use random greedy search to select specific features from the 15 feature templates defined in Section 3.4.1. Features seen fewer than three times in the development are thrown out. Separate regression models are built for three different parsers. The logistic regression classifier uses an L_1 penalty with regularization parameter $C = 1$.

[15] Decreasing m did not lead to further improvement and thus we don't report the results of changing m.

[16] We only use sentences with different scores to limit ceiling effects.

	Avg	BNC	Brown	QB	WSJ
RBG	86.4	89.2	90.9	75.8	93.7
+ S-Dual	86.8	89.8	90.9	**76.5**	**94.0**
+ Logit	**86.9**	**89.8**	**91.1**	**76.5**	**94.0**
BLLIP	89.5	91.1	91.6	83.3	94.2
+ Pair	90.0	**91.4**	**92.3**	84.1	94.1
+ Logit	**90.3**	91.3	92.1	**85.2**	**94.3**
BLLIP-ST	90.1	92.7	92.3	84.3	93.8
+ Pair	90.7	**93.5**	92.5	85.6	93.8
+ Logit	**91.1**	93.3	**92.6**	**86.7**	**93.9**

Table 6: Effect of using logistic regression on top of each method (UAS). Leave-one-out cross-validation is performed on the development data. +X means augmenting the above system with X.

Logistic regression experiments are reported in Table 6. All parsers benefit from employing logistic regression models on top of paraphrase methods. BLLIP experiments show a larger improvement than RBG. This may be because BLLIP cannot use soft constraints so its errors are more pronounced.

5.4 Final Evaluation

We evaluate the three parsers on the test set using the tuned parameters and logistic regression models from above. Joint parsing with paraphrases significantly improves accuracy for all systems (Table 7). Self-trained BLLIP with logistic regression is the most accurate, though RBG with S-Dual provides the most consistent improvements.

Joint parsing without logistic regression (RBG + S-Dual) is more accurate than independent parsing (RBG) overall. With the help of logistic regression, the methods do at least as well as their baseline counterparts on all domains with the exception of self-trained BLLIP on BNC. We believe that the drop on BNC is largely due to noise as our BNC test set is the smallest of the four. As on development, logistic regression does not change the accuracy much over the RBG parser with soft dual decomposition.

Joint parsing provides the largest gains on QuestionBank, the domain with the lowest baseline accuracies. This fits with our goal of using paraphrases for domain adaptation — parsing with paraphrases helps the most on domains furthest from our training data.

5.5 Error analysis

We analyzed the errors from RBG and BLLIP along several dimensions: by dependency label, sentence length, dependency length, alignment status (whether a token was aligned), percentage of tokens aligned in the sentence, and edit distance between the sentence pairs. Most errors are fairly uniformly distributed across these dimensions and indicate general structural improvements when using paraphrases. BLLIP saw a 2.2% improvement for the ROOT relation, though RBG's improvement here was more moderate. For sentence lengths, BLLIP obtains larger boosts for shorter sentences while RBG's are more uniform. RBG gets a 1.4% UAS improvement on longer dependencies (6 or more tokens) while shorter dependencies are more modestly improved by about 0.3-0.5% UAS. Surprisingly, alignment information provides no signal as to whether accuracy improves.

Additionally, we had our annotator label a portion of our dataset with the set of paraphrasing operations employed.[17] While most paraphrasing operations generally improved performance under joint inference, the largest reliable gains came from lexical replacements (e.g., synonyms).

6 Conclusions and Future Work

Our methods of incorporating paraphrases improve parsing across multiple domains for state-of-the-art constituency and dependency parsers. We leverage the fact that paraphrases often express the same semantics with similar syntactic realizations. These provide benefits even on top of self-training, another domain adaptation technique.

Since paraphrases are not available at most times, our methods may seem limited. However, there are several possible use cases. The best case scenario is when users can be directly asked to rephrase a question and provide a paraphrase. For instance, question answering systems can ask users to rephrase questions when an answer is marked as wrong by users. Another option is to use crowdsourcing to quickly create a paraphrase corpus (Negri et al., 2012; Burrows et al., 2013; Tschirsich and Hintz, 2013). As part of future work, we plan to integrate existing larger paraphrase resources, such as WikiAnswers (Fader et al., 2013) and PPDB (Ganitkevitch et al., 2013). WikiAnswers provides rough equivalence classes

[17] See Appendix A for more information about this task and its results.

	Avg	BNC	Brown	QB	WSJ
RBG	86.7 (81.3)	89.3 (83.7)	90.2 (84.1)	77.0 (71.0)	93.7 (89.9)
+ S-Dual	**87.3 (81.7)**	89.6 (83.8)	**90.7 (84.6)**	**78.1 (71.8)**	**94.0 (90.2)**
+ Logit	87.2 (81.6)	**89.7 (83.9)**	90.6 (84.5)	77.9 (71.7)	93.8 (89.9)
BLLIP	89.6 (86.1)	90.6 (87.2)	91.7 (87.9)	83.6 (79.9)	94.3 (91.6)
+ Pair	90.1 (86.5)	**90.8 (87.3)**	**92.1 (88.4)**	84.7 (80.7)	94.4 (91.6)
+ Logit	**90.3 (86.8)**	90.6 (87.2)	91.9 (88.1)	**85.5 (81.7)**	**94.5 (91.7)**
BLLIP-ST	90.4 (87.0)	**91.8 (88.3)**	92.7 (89.0)	84.8 (81.2)	**94.3 (91.4)**
+ Pair	90.5 (87.1)	91.1 (87.6)	92.7 (89.1)	85.5 (81.8)	94.2 **(91.4)**
+ Logit	**91.0 (87.6)**	91.4 (88.0)	**92.9 (89.3)**	**86.6 (82.9)**	**94.3 (91.4)**

Table 7: Final evaluation on testing data. Numbers are unlabeled attachment score (labeled attachment score). +X indicates extending the above system with X. BLLIP-ST is BLLIP using the self-trained model. Coloring indicates a significant difference over baseline ($p < 0.01$).

of questions. PPDB includes phrasal and syntactic alignments which could supplement our existing alignments or be used as proxies for paraphrases. While these resources are noisy, the quantity of data may provide additional robustness. Lastly, integrating our methods with paraphrase detection or generation systems could help provide paraphrases on demand.

There are many other ways to extend this work. Poor alignments are one of the larger sources of errors and improving alignments could help dramatically. One simple extension is to use multiple paraphrases and their alignments instead of just one. More difficult would be to learn the alignments jointly while parsing and adaptively learn how alignments affect syntax. Our constraints can only capture certain types of paraphrase transformations currently and should be extended to understand common tree transformations for paraphrases, as in (Heilman and Smith, 2010). Finally, and perhaps most importantly, our methods apply only at inference time. We plan to investigate methods which use paraphrases to augment parsing models created at train time.

Acknowledgments

We would like to thank Eugene Charniak for the idea of using paraphrases to improve parsing, our anonymous reviewers for their valuable feedback, Karen Ingraffea for constructing and classifying the paraphrase corpus, Dave Buchanan and Will Headden for last minute paper reading, the DeepQA team at IBM for feedback and support on the project, and Mohit Bansal and Micha Elsner for helpful discussions.

References

Sam Anzaroot, Alexandre Passos, David Belanger, and Andrew McCallum. 2014. Learning soft linear constraints with application to citation field extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 593–602. Association for Computational Linguistics.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer speech & language*, 20(1):41–68.

Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 693–702. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425. Association for Computational Linguistics.

Rahul Bhagat and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–135. Association for Computational Linguistics.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):43.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the main*

conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pages 17–24. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2013. An abstractive approach to sentence compression. *ACM Transactions on Intelligent Systems and Technology*, 4(3):41.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 175–182. ICML.

Gregory Coppola and Mark Steedman. 2013. The effect of higher-order dependency features in discriminative phrase-structure parsing. In *ACL*, pages 610–616.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, pages 449–454.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 33–36. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics*, pages 340–345. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1608–1618. Association for Computational Linguistics.

Jennifer Foster and Josef van Genabith. 2008. Parser evaluation and the BNC: Evaluating 4 constituency parsers with 3 metrics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. European Language Resources Association.

Winthrop Nelson Francis and Henry Kučera. 1989. *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764. Association for Computational Linguistics.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

John Judge, Aoife Cahill, and Josef Van Genabith. 2006. QuestionBank: creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 497–504. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL: HLT*, pages 595–603. Association for Computational Linguistics.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1381–1391. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

Preslav Nakov and Marti Hearst. 2005. Using the web as an implicit training set: Application to structural ambiguity resolution. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 835–842, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Matteo Negri, Yashar Mehdad, Alessandro Marchetti, Danilo Giampiccolo, and Luisa Bentivogli. 2012. Chinese whispers: Cooperative paraphrase acquisition. In *LREC*, pages 2659–2665.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Alexander M. Rush and Michael Collins. 2012. A tutorial on dual decomposition and Lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45(1):305–362.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1434–1444. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Martin Tschirsich and Gerold Hintz. 2013. Leveraging crowdsourcing for paraphrase recognition. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 205–213, Sofia, Bulgaria, August. Association for Computational Linguistics.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 656–661. Association for Computational Linguistics.

Op	Meaning
S	syntax-preserving lexical changes (e.g., singular noun → other singular noun, singular noun → plural noun, "enunciated" → "said", "their" → "the", "which" → "that", includes cases 1, 11, 12, and 14 in Bhagat and Hovy (2013))
D	simple deletion (with minimal changes to surrounding syntax)
I	simple insertion (with minimal changes to surrounding syntax)
P	small phrase substitution (phrase doesn't change syntactic type, e.g., "elephant" → "big animal")
M	movement (phrase moved, punctuation fixed up, but no other changes, e.g., topicalization)
C	lexical conversion *or* verb conjugation changes which change syntax like "speak" → "speaking", possibly with changes to surrounding words (e.g., noun → verb, includes cases 15–17 and 21 in Bhagat and Hovy (2013))
O	other more disruptive transformations, not classifiable above

Table 8: Description of paraphrase operations.

A Paraphrase operations

In addition to creating the paraphrase dataset (Section 4), we had the same linguist annotate 696 randomly selected sentence pairs from the dataset. Each sentence pair was annotated with the set of paraphrase operations employed to transform the original sentence into the paraphrase. Paraphrase operation codes and their descriptions are shown in Table 8, ordered roughly from lowest to highest expected impact on the syntactic tree. Changes to punctuation was not considered for these operations. For pairs which didn't fit cleanly into any of the paraphrase operations listed or in cases where they couldn't be easily discerned, the pair was labeled only as 'O' (other). This was largely done to simplify the task since it can be difficult to determine the boundaries between operations. Additionally, for operations that imply others (e.g., S → I + D), only the more complex operation is listed (S, in this case) unless its component operations show on their own as well. The paraphrase operations in Table 8 are not exhaustive (e.g., many described in Bhagat and Hovy (2013) are not included), but cover a significant portion of the operations employed in our dataset. The list of operations was created and refined during a pilot paraphrase classification by the authors to reflect the operations typically used in the dataset.

A.1 Analysis

Table 9 lists various statistics about the paraphrase operations employed in the dataset. In Table 9(a), we provide the distribution of all operations employed across the dataset. A range of different paraphrase operations are used. Movement, phrase, and lexical substitution are the most frequent classifiable operations employed, perhaps because they are the simplest to apply. Note these counts are the counts of individual operations and a single sentence pair may employ multiple operations. This is why the percent for O in this table does not match up with Tables 9(b) and 9(c) which count operations at the sentence pair level.

Table 9(b) gives the counts for how many operations were employed at the sentence pair level (e.g., 150 sentence pairs use exactly two paraphrase operations from Table 8). Nearly 60% of the sentence pairs were able to be cleanly classified. For the classifiable pairs, most employed only one operation, though a significant portion used a pair of operations (the nature of which is illustrated in the next table).

In Table 9(c), we give the distribution of frequent paraphrase operation combinations at the sentence pair level. The most common combination which employs multiple (classifiable) operations is S+M (substitution and movement). Movement is quite common as a secondary operation and appears in a pair with all the other operations (I+M is substantially more common than I on its own).

Finally, Table 9(d) lists the distribution of paraphrase operations by domain. The distribution is roughly similar across domains but there are some differences — BNC employs C (conversion/verb conjugation) roughly twice as frequently as the other domains. QuestionBank has the highest number of unclassifiable operations (O). Many of these are rephrasing questions to put them in their "Jeopardy!" form: "How many member states are in the UN?" → "The UN has this many member states."

(a) Individual operation counts

Operation	Count	Percent
O	296	31.4%
M	194	20.6%
P	151	16.0%
S	114	12.1%
I	73	7.7%
D	65	6.9%
C	49	5.2%

(b) Number of operations used

Number	Count	Percent
1	206	29.6%
2	150	21.6%
3	36	5.2%
4	8	1.1%
O	296	42.5%

(c) Combined operation counts

Operations	Count	Percent
O	296	42.5%
P	86	12.4%
M	57	8.2%
S	43	6.2%
S M	31	4.5%
PM	29	4.2%
I M	22	3.2%
MC	16	2.3%
S P	10	1.4%
DI M	8	1.1%
C	8	1.1%
DI	7	1.0%
D M	7	1.0%
SD	6	0.9%
D C	6	0.9%
D	6	0.9%
I	6	0.9%
24 others (count ≤ 5)	52	7.5%

(d) Operation distribution by domain (%)

Op	BNC	Brown	QB	WSJ
S	16.9	21.7	9.5	17.9
D	10.4	14.4	4.8	8.1
I	13.0	11.1	6.9	11.6
P	17.5	21.1	20.6	27.2
M	32.5	28.9	15.9	35.8
C	12.3	6.1	5.3	5.2
O	40.9	34.4	56.1	37.6

Table 9: Paraphrase operation statistics.

	UAS						LAS					
Op	BLLIP	+ Logit	Δ	RBG	+ Logit	Δ	BLLIP	+ Logit	Δ	RBG	+ Logit	Δ
S	91.9	93.0	+1.1	90.4	91.5	+1.1	88.4	89.3	+0.9	85.0	85.8	+0.8
D	93.9	94.1	+0.2	91.3	91.8	+0.4	90.4	90.4	—	85.8	85.9	+0.1
I	93.4	93.8	+0.4	90.1	90.5	+0.4	89.2	89.4	+0.2	83.9	84.0	+0.1
P	89.2	90.2	+1.0	88.3	88.3	—	85.9	86.9	+1.0	82.7	82.6	-0.1
M	92.5	92.9	+0.4	89.2	89.6	+0.4	88.8	89.1	+0.3	83.5	83.9	+0.4
C	91.0	91.3	+0.3	89.5	90.4	+0.9	86.7	86.8	+0.1	83.1	84.0	+0.9
O	90.4	90.7	+0.3	86.6	87.1	+0.5	87.1	87.5	+0.4	81.4	81.7	+0.3

Table 10: Impact of paraphrasing operations on joint parsing performance with BLLIP and RBG.

A.2 Operation impact on joint parsing performance

In addition to better understanding the composition of paraphrase operations in our dataset, we are interested in whether certain paraphrase operations improve joint parsing performance more than others. We can get a rough sense of this by grouping all sentences which employ a particular operation and scoring their parses from their base parser (BLLIP or RBG) against its joint counterpart (BLLIP with pair finding and logistic regression or RBG with soft dual decomposition and logistic regression). There are two caveats, though: (1) These divisions result in relatively small sets of sentence pairs, so these differences may not be reliable and (2) This analysis assumes that paraphrase operations are independent which they clearly aren't. One possible direction for future work would be to create a larger paraphrase dataset where each sentence pair only employed a single phrase operation.

Keeping these issues in mind, we list the results of our analysis in Table 10. For BLLIP, most operations provide some benefits to both UAS and LAS, though lexical and phrase substitutions provide the largest gains. RBG is similar, but does not see the same gains from phrase substitutions. It does obtain a large benefit from conversions, but since these are the rarest operation, its full impact cannot be determined from our classifications.

B Example paraphrases

We list three randomly selected sentence pairs from each corpus in our dataset. The upper sentence in each pair is the original and the lower sentence is the paraphrase made by our annotator.

B.1 British National Corpus

Richmann looked over at the steam yacht, and made a circling motion with his hand. Making a circling motion with his hand, Richmann looked at the steam yacht.
He nodded his thanks as he took it and flipped another cigarette butt into the toilet. Flipping another cigarette butt into the toilet, he took it and nodded his thanks.
The longer spines are mace-like; with a slightly rugose body and a head with multiple points. There are mace-like longer spines, a slightly rugose body, and a multi-pointed head.

B.2 Brown

The men behind them were Bill Doolin and five of his gang – every man a killer. Every man behind him was a killer: Ben Doolin and five of his gang.
She musta been walking in her sleep – you seen her yourself in here". "But you saw her there yourself, she must have been sleepwalking."
The record teems with romance and adventure. Romance and adventure abound in the record.

B.3 QuestionBank

How do you say “fresh” in Spanish? What is the Spanish word for “fresh”?
Who was Garrett Morgan married to? Who did Garrett Morgan marry?
What book does Holden Caulfield appear in? Holden Caulfield is a character in what book?

B.4 WSJ

Before the halt, AMR last traded at 98 5/8. AMR was trading at 98 5/8 before the halt.
Sequa makes and repairs jet engines. Sequa is in the business of making and repairing jet engines.
The Big Board also added computer capacity to handle huge surges in trading volume. Computer capacity was also added by the Big Board, in order to handle huge surges in trading volume.