Generating Disambiguating Paraphrases for Structurally Ambiguous Sentences

Manjuan Duan and **Ethan Hill** and **Michael White**
Department of Linguistics
The Ohio State University
Columbus, OH 43210, USA
{duan,mwhite}@ling.osu.edu, hill.1303@gmail.com

Abstract

We present a method that, for the first time in a broad coverage setting, uses natural language generation to automatically construct disambiguating paraphrases for structurally ambiguous sentences. By simply asking naive annotators to clarify which paraphrase is closer in meaning to the original sentence, the resulting paraphrases can potentially enable meaning judgments for parser training and domain adaptation to be crowd-sourced on a massive scale. To validate the method, we demonstrate that meaning judgments crowd-sourced in this way via Amazon Mechanical Turk have reasonably high accuracy—e.g. 80%, given a strong majority choice between two paraphrases—with accuracy increasing as the level of agreement among annotators increases. We also show that even with just the limited validation data gathered to date, the crowd-sourced judgments make it possible to retrain a parser to achieve significantly higher accuracy in a novel domain. We conclude with lessons learned for gathering such judgments on a much larger scale.

1 Introduction

While early dialogue systems such as SHRDLU (Winograd, 1973) were capable of asking questions to clarify the meaning of structurally ambiguous sentences, to our knowledge the task of generating questions to clarify structural ambiguities has not been investigated on a broad scale. Given the development in recent years of statistical parsers and realizers using a reversible grammar or a common set of dependencies, one might expect that in principle it should be possible today to generate paraphrases to help clarify the meaning of structurally ambiguous sentences simply by chaining the parser and realizer end-to-end. However, realization ranking models are typically trained to prefer corpus sentences over possible variants, and thus statistical realizers chained with statistical parsers are apt to just reproduce the input sentence, which is of no help for disambiguation. Moreover, while it is easy enough to require the realizer to output a distinct sentence, for most realizers there is no guarantee that the realization will in fact unambiguously express one or the other possible meaning.

In early work in natural language generation, Neumann and van Noord (1992) investigated algorithms for avoiding ambiguity in surface realization. More recently, Duan and White (2014) developed a method for using statistical parsers together with a realization ranking model to balance the competing concerns of fluency and ambiguity avoidance, given that sentences of even moderate length are rarely unambiguous according to a broad coverage grammar. In this paper, we present and validate a related method that aims to ensure that the difference in dependencies between two competing parses is unambiguously expressed in the realization corresponding to each parse (albeit at the expense of fluency), so that the realizations can serve as disambiguating paraphrases for the input sentence. To the extent that the method is successful, it then becomes possible to clarify the meaning of structurally ambiguous sentences simply by asking naive annotators which paraphrase is closer in meaning to the original sentence.

As is well known, the performance of most NLP tools such as statistical parsers has remained much higher for the domains and genres for which large-scale annotated training corpora are available. Domain adaptation techniques are not always successful (Dredze et al., 2007), and while

Proceedings of LAW X – The 10th Linguistic Annotation Workshop, pages 160–170,
Berlin, Germany, August 11, 2016.

self-training can yield substantial error reductions (McClosky and Charniak, 2008; Honnibal et al., 2009), large gaps in performance persist. Consequently, to achieve high performance, there remains a need to collect new annotated data in the target domain and genre. Moreover, experience with ImageNet (Deng et al., 2009; Russakovsky et al., 2015) in vision research suggests that breakthroughs in NLP performance might likewise be enabled by collecting annotated data across domains and genres on a massive scale.

As a first step towards that end, we present a validation experiment which demonstrates that our method enables meaning judgments to be crowdsourced on Amazon's Mechanical Turk (AMT) with reasonably high accuracy, achieving 80% agreement with our own gold standard judgments when there is a strong majority choice between two paraphrases. Moreover, accuracy remains satisfactorily high for the subset of sentences where the top parse is incorrect. We also present a preliminary experiment which shows that even with just the limited validation data gathered to date, the "silver standard" crowd-sourced judgments make it possible to retrain a parser to achieve significantly higher accuracy in a novel domain.

In a previous study on obtaining crowd-sourced syntactic annotations, Jha et al. (2010) presented results indicating that with some training, annotators on AMT could accurately select prepositional phrase (PP) attachment sites, with accuracy also increasing with the level of agreement among annotators. Gerdes (2013) and Zeldes (2016) also found that it was possible to obtain fairly high quality class-sourced annotations where students only received a modest amount of training. Our work is quite different in that we aim to gather meaning judgments with no training whatsoever, simply by asking questions in natural language. Our work also differs from Jha et al.'s in that it is not limited to PP-attachment ambiguities. Since the Jha et al. study used a different corpus, our results are not directly comparable, though we note that our method also achieves satisfactory accuracy on PP-attachment cases. Finally, we note that our paper shows that the crowd-sourced data can enable parser improvements, while their study does not include parser retraining results.

The paper is structured as follows. In Section 2, we review the parsing and realization ranking models that serve as a starting point for the paper. In Section 3, we present our method for generating disambiguating paraphrases. In Section 4, we present our experiment validating the accuracy of naive annotator choices on AMT. In Section 5, we present an analysis of errors and a regression analysis investigating the factors affecting annotator decisions. In Section 6, we present our preliminary parser retraining experiment. In Section 7, after briefly comparing our results with Jha et al.'s, we discuss the implications of the analyses for future data collection and parser adaptation experiments. Finally, in Section 8, we conclude with a summary of the lessons learned.

2 Background

To generate disambiguating paraphrases, we use OpenCCG, an open source framework for parsing and realization with Combinatory Categorial Grammar (Steedman, 2000). It comes with a broad coverage English grammar extracted from a version of the CCGbank (Hockenmaier and Steedman, 2007) enhanced to include (inter alia) assignment of consistent semantic roles across diathesis alternations (Boxwell and White, 2008), using PropBank (Palmer et al., 2005). The parser can be used with a reimplemented version of Hockenmaier & Steedman's (2002) generative model or with the Berkeley parser (Petrov et al., 2006; Fowler and Penn, 2010); in this paper we use the Hockenmaier & Steedman model. The outputs of the parser—and the inputs to the realizer—are semantic dependency graphs, or logical forms, examples of which are given in the next section. In these graphs, nodes correspond to discourse referents labeled with lexical predicates and semantic attributes, and dependency relations between nodes encode argument structure.

The realizer uses a chart-based algorithm (White, 2006) together with a "hypertagger" for probabilistically assigning lexical categories to lexical predicates in the input (Espinosa et al., 2008). To select preferred outputs from the chart, we use an averaged perceptron realization ranking model (White and Rajkumar, 2009) that combines Clark & Curran's (Clark and Curran, 2007) normal-form syntactic model and various n-gram models including a large-scale 5-gram model based on the Gigaword corpus together with a feature for dependency length minimization (White and Rajkumar, 2012) and features for enhanced syntactic agreement (Rajkumar and White, 2010).

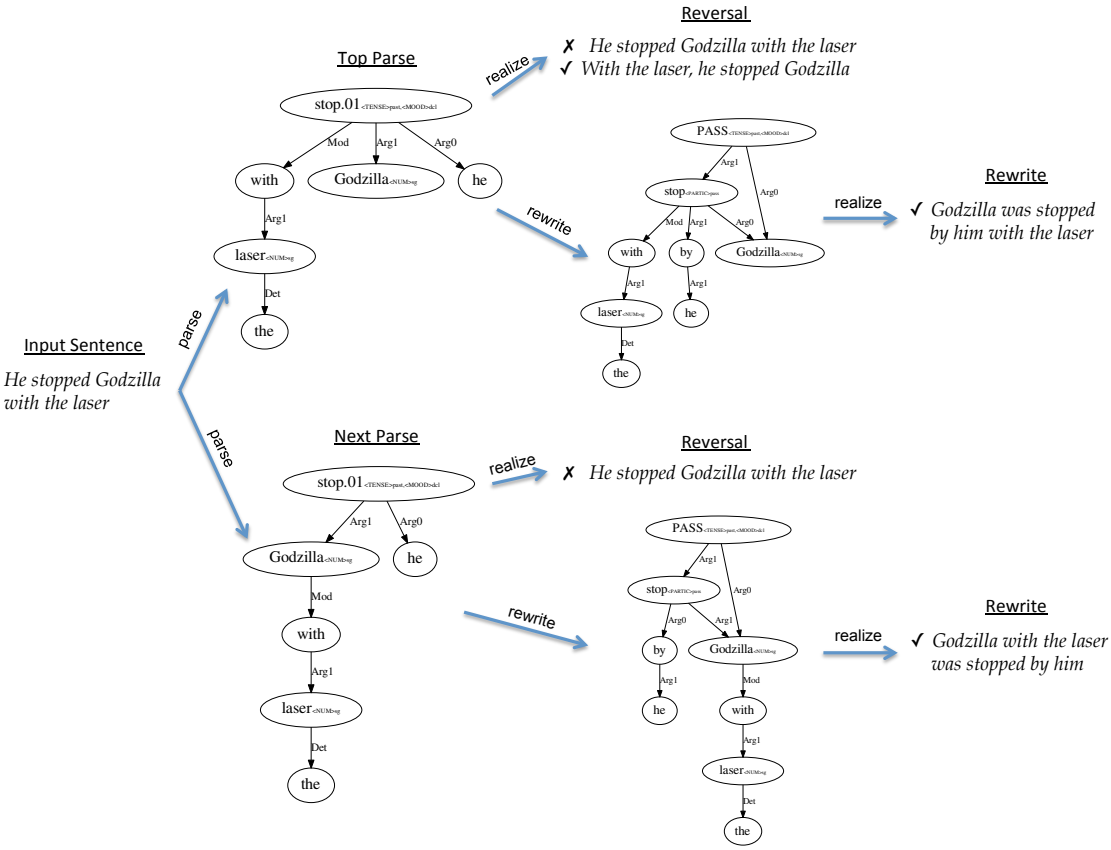

Figure 1: Overview of paraphrasing process (see text)

3 Generating Disambiguating Paraphrases

3.1 Parsing

At an overview level, the process for automatically generating disambiguating paraphrases is shown in Figure 1.[1] The first step is to obtain n-best parses of the input sentence (with $n = 25$ in our experiments).[2] Any structurally broken parses, such as those with two roots, are filtered out. Next, the remaining parses are examined successively to determine whether there is a parse that is *sufficiently distinct* from the top parse so that paraphrases generated from these two parses can be meaningfully distinguished. In order to locate a meaningful difference between two parses, the unlabeled and unordered dependencies extracted from the the top parse and a parse from the n-best parse list are compared. To be considered sufficiently distinct, the symmetric difference between the simplified dependencies must be non-empty, with neither set of dependencies a superset of the other, so that the difference between the parses represents a distinct attachment decision. For example, ambiguities involving only POS, named entity or word sense differences are not considered sufficiently distinct. If successful, this phase yields a *top* and *next* parse, whose distinct dependencies indicate the meaning difference for which the parser has the greatest uncertainty, given the relatively high probabilities assigned to both interpretations.

3.2 Reverse Realizations

Once the top and next parses have been selected, the next step is to realize the two distinct parses into their respective surface realizations, choosing the realizations which meet the criteria listed below for being disambiguating paraphrases of the original sentences. Paraphrases obtained in this process are called *reversals* in our study. Specifically, each parse is realized back into a n-best re-

[1] The input sentence *He stopped Godzilla with the laser* is one of the simplest in our test domain of Wikipedia articles on prehistoric reptiles, which contains occasional references to such creatures appearing in popular media.

[2] Although it is possible that some parses which represents meaningful structure differences might fall outside of the top 25 parses, we choose n to be 25 because the quality of parses generally goes down quickly when moving down the list.

alization list (with $n = 25$), which is traversed in order to find a qualifying paraphrase. The first criterion is that the realization needs to be different from the original sentence to be qualified as a paraphrase; in Figure 1, such non-helpful exact matches are crossed off. However, not just any realization that differs from the original sentence is necessarily disambiguating: it may just have a minor change in a part of the sentence unrelated to the ambiguity in question. Thus, we define the relevant *ambiguity span* for the sentence, and ensure that this span is altered in the realization.

Take a sentence from *Prehistoric Reptiles* corpus as an example.

(1) The two adult T-Rex and their baby are shown to have been returned safely.

Here the unlabeled dependency set of the top parse contains the dependency *returned* $\rightarrow$ *safely*, while a parse down in the n-best list has the dependency *shown* $\rightarrow$ *safely*. The three words *shown* $<$ *returned* $<$ *safely* form an ambiguity span in the original sentence for this ambiguity. When selecting the paraphrases from the n-best realizations, we choose the realization which has different relative distances for the words involved in the ambiguity span in the original sentence. For the dependency *returned* $\rightarrow$ *safely*, the realization *The two adult T-Rex and their baby are* ***shown*** *to have been* ***safely returned***, in which the relative distances between *shown*, *safely* and *returned* are changed, is selected. In the same way, for the other dependency *shown* $\rightarrow$ *safely*, the realization *The two adult T-Rex and their baby are* ***shown safely*** *to have been* ***returned*** is selected. The two realizations are then parsed to verify that the most likely interpretation does include the two dependencies from which they are generated. By doing so, we want to make sure that the realizations are structurally representative of the meaning for which it is chosen. If they pass the verification, these two realizations will be selected as the two paraphrases of the original sentence, each paraphrase representing a possible interpretation of the original sentence. These two paraphrases are called *two-sided* paraphrases of the original sentence.

In some cases, we fail to find two paraphrases of the original sentence, i.e. the algorithm fails to find a sentence in the n-best realizations of one of the distinct parses which is different from the original sentence, breaks up the ambiguity span and passes the verification. In these cases, we only generate one paraphrase for the original sentence, with the assumption that the other interpretation of the sentence is expressed by the original sentence. We call these cases *one-sided* paraphrases here.

3.3 Logical Form Rewrites

As noted above, there are some cases where it is impossible to generate a reversal that expresses one of the possible interpretations of the original sentence without repeating the original sentence. For example, the sentence *He stopped Godzilla with the laser* is ambiguous about whether the prepositional phrase *with the laser* is modifying *Godzilla* or the verb *stop*, as shown in Figure 1. It is impossible to have a reversal which expresses the interpretation where the prepositional phrase is modifying *Gozilla* and where the ambiguity span is altered, as the figure shows. In cases like these, we force structure changes in the dependency graphs, which, when realized, can demonstrate the parse's interpretation more adequately. The resulting realizations are referred to as *rewrites*.

Specifically, we experiment with three types of logical form rewriting: passive rewrites, cleft rewrites and coordination rewrites. Passive and cleft rewrites are designed for PP-attachment ambiguities, while coordination rewrites are for ambiguities in the scope of modifiers with coordinated phrases.[3]

For passive and cleft rewrites, we first detect the presence of a PP-attachment ambiguity by examining the POS tags of the dependents involved in the ambiguous dependencies. If we find the same prepositional phrase is attaching to different heads in distinct parses, we regard this as a PP-attachment ambiguity case. We then examine the main verb of the sentence to make sure the verb can be passivized or clefted. To force a passive rewrite, we create a passive node with the same tense as the original sentence and make the Arg1 of the main verb the Arg0 of the new node and attach the main verb to the passive node as a complement. The original Arg0 is replaced by a prepositional phrase *by Arg0* attached to the main verb, as illustrated in Figure 1.

For cleft rewrites (not shown in the figure), we create a *be* verb node, with the same tense as the original sentence, above the main verb of the clause containing the PP-attachment ambigu-

[3]These ambiguities are frequently found to be involved in the errors of most parsers; we leave experimenting with other kinds of rewrites for future work.

ity. We then create a *what* reference node taking the whole verb phrase as its complement and attach the *what* reference node to the verb *be* as a complement, yielding for example *Godzilla with the laser was what he stopped.*[4]

A coordination ambiguity refers to the cases where a modifier can be modifying the first conjunct or modifying the whole conjoined phrase, e.g. the modifier *East/West* in *He also was selected to play in [the] East/West Shrine game and Hula bowl.*[5] For the parse in which the modifier modifies the first conjunct only, we swap the order of the two conjuncts, so the conjunct with a modifier will occur after the conjunction, as in *He also was selected to play in [the] Hula bowl and [the] East/West Shrine game.* In the case where the modifier is modifying the whole conjoined phrase, we force verbosity in the logical form by moving the modifier to each conjunct and then swap the order of the two conjuncts, as in *He also was selected to play in [the] East/West Hula bowl and [the] East/West Shrine game.*

4 Validation Experiment

4.1 Data

We collected 6,335 sentences from *Prehistoric Reptiles* and 7,779 from *Big 10 Conference Football* from English Wikipedia. Only sentences with length of 5 to 20 words were selected to parse, assuming simple sentences would generalize better for parser adaptation. After parsing these sentences, for 2,458 sentences (38.8% of total sentences) from *Prehistoric Reptiles* and 2,605 sentences (33.5% of total sentences) on *Big 10 Conference Football*, we found meaningfully distinct parses in their n-best parse list. Of these 5,063 sentences, valid paraphrases are generated for 3605 sentences (71.2% of 5,063).

From these sentences, we randomly chose 515 sentences from each domain to be our test set, weighted to favor two-sided cases. In these 1030 sentences, 75% are two-sided cases and 25% are one-sided cases; 65% are reversals and 35% are rewrites (15% from cleft rewrites, 15% from coordination rewrites, and 10% from passive rewrites).

[4]The *what*-node is actually underspecified between *what* and *who(m)*, leaving the realizer to make the choice.

[5]This sentence, from our test domain of Big 10 Football, is mistakenly missing the determiner *the* in Wikipedia.

4.2 Annotation

For the 1030 sentences, we decided on the optimal ('gold') interpretation of the disputed dependencies represented by the two distinct parses. We annotated the correct parse by examining the dependency graphs. If the top parse was correct in the ambiguous dependency, the sentence was annotated as 'top'. A sentence was annotated as 'next' if the next best parse was correct in terms of the disputed dependencies. When neither of the two parses was more correct than the other one (e.g., when neither parse had the correct PP-attachment), the sentence was annotated as 'neither'; this label also covered some cases where there was no discernible semantic difference between the cases.

100 sentences were triple-annotated; for these sentences, inter-annotation agreement was **82.5%** for all three labels and **90.8%** excluding the 'neither' cases. The remaining sentences were single-annotated, with discussion of difficult cases. Of the 1030 sentences, 56.3% were annotated as top, 25.4% were 'next' and 18.3% were 'neither' cases. To calculate accuracy of Turker judgments below, we excluded the 'neither' cases; however, we included them for data collection since in a typical (non-validation) data collection scenario, the identity of the 'neither' cases would not be known.

4.3 Judgment Collection

For each of the 1030 sentences, we collected 5 judgments from the workers on Amazon Mechanical Turk. For each sentence, we provided a comprehension question to prevent random choosing; accuracy on comprehension questions was 93%, indicating that workers were paying attention to the task. For the sentences with two paraphrases, we asked the worker to choose which out of these two was closer to the original sentence in terms of meaning. For the one-sided cases, we simply asked them to decide whether that paraphrase had the same meaning as the original sentence.

We put 25 sentences into each survey and paid $2 per survey. It took around 20 minutes on average to finish a survey. In total, we paid $400 for 5000 judgments from AMT workers. While it took the authors days to come up with the gold annotations by examining the parses, the AMT judgments were collected in just a few hours.

	Maj	S. Maj	Unani
Coverage	99.3	69	36
Accuracy	68.1	76.4	82.6

Table 1: Coverage and Accuracy

	Maj	S. Maj	Unani
One-sided	59.1	65.2	70.6
Two-sided	71	79.9	87.2
Reversals	69.3	79.9	88.2
Rewrites	74.8	79.8	84.6
Cleft	79.7	82.1	83.3
Passive	68	71.4	66.6
Coordination	70	79.2	88

Table 2: Accuracy of AMT workers' judgments

4.4 Results

Table 1 shows the trade-off between the accuracy of the judgments collected from AMT and the coverage of the data. In Table 1, sentences which have more than 50% agreement from AMT workers are called 'Majority' cases (Maj); those with more than 75% agreement are 'Strong Majority' cases (S. Maj) and those with 90% or more agreement are 'Unanimity' cases (Unani).[6] As the table shows, the 'unanimity' sentences have the highest accuracy, however, at the expense of losing the coverage of 64% data.

Table 2 shows the accuracy of the AMT workers' judgments under different settings. The results shown in Table 2 are all significantly better than random choice (p = 0.5) at a level $\alpha = 0.05$ (binomial sign test). Table 2 shows that two-sided paraphrases have considerably higher accuracy than one-sided cases, which means two-sided paraphrases are better in highlighting the ambiguity in the original sentence.

Table 2 also shows the accuracy of reversals and rewrites for the two-sided paraphrases. It is good to see that reversals work better than rewrites in strong majority cases and unanimity cases, because reversals can be obtained without any changes to the logical forms and are able to capture various kinds of structural ambiguities detected by the automatic parser, not just those the rewrites have been designed to capture. 'Strong majority' two-sided cases appear to offer the best balance between coverage and accuracy.

[6]There are a few duplicated sentences in the validation dataset. For each of these sentences, we might have 10 or 15 Turker judgments. As such, we define 'Strong Majority' as agreement more than 75% and 'Unanimity' as agreement more than 90%.

	Maj	S. Maj	Unani
Total	**59.6**	**68**	**74.6**
One-sided	49.1	53.5	70.6
Two-sided	**63.2**	**73.9**	**87.2**
Reversal	55.3	**66.3**	**75.9**
Rewrite	**67.5**	**70.5**	**85.7**
Cleft	**81.8**	**86.6**	85.7
Passive	58.8	57.1	62.5
Coordination	63.2	**68.8**	**78.6**

Table 3: Accuracy of 'next' parses (accuracies significantly higher than chance in bold)

In order to judge whether the crowd-sourced judgments can be potentially beneficial for parser retraining, we need to examine the proportion of 'next' cases (i.e., those sentences one of whose non-top parses is more accurate than the top parse) that can be correctly annotated. Table 3 shows that majority, strong majority and unanimous annotations are all significantly better than chance overall in these cases ($p \leq 0.05$, exact binomial test). Some of the individual results in Table 3, however, fail to reach the significance level because of the small sample sizes. For example, the unanimous annotations for cleft are correct in 6 sentences out of 7 sentences; although the accuracy is as high as 85.7%, it still fails to be significant because of the small sample size. In general, two-sided paraphrases still work better than the one-sided ones and rewrites work better than reversals in terms of correctly annotating 'next' cases.

5 Error Analysis

5.1 Manual analysis

We did not directly evaluate paraphrase quality in this study, as we were primarily concerned with whether they sufficed to enable accurate crowd-sourced judgments. However, we did manually analyze 43 sentences where the unanimous AMT worker judgments do not agree with the expert annotations and found the following reasons: incompetent or broken realizations (29 out of 43); bad parses (11 out of 43); lack of context (3 out of 43).

Incompetents realizations refer to those paraphrases which fail to convey the distinct meanings in the parses in a distinguishable way. Sometimes a change of adverbial position in a sentence or punctuation deletion/insertion does not alter a

human reader's interpretation of the sentence. For example, in (2) below, (2a) is the original sentence, which is ambiguous as to whether *with* attaches to the verb *crush*, which is realized as (2c), or to the noun *animal*, whose realization is the same as the original sentence. The correct interpretation is that *with* attaches to *animals*, so the expert annotation is (2b). Compared with the original sentence, (2c) has a comma inserted after *animals*. However, all 5 AMT workers think (2c) has the same meaning as the original sentence in spite of this change, as the punctuation difference is too subtle for reliable interpretation.

(2) a. The teeth were adapted to *crush* bivalves, gastropods and other *animals* **with** a shell or exoskeleton.
b. (animals→with): *Same as original sentence*
c. (crush→with): The teeth were adapted to crush bivalves, gastropods and other animals**,** with a shell or exoskeleton.

In some cases, the AMT workers fail to choose the correct parse because the realization of the correct parse is much less fluent than the other one. In (3) below, (3a) is the original sentence, and it is ambiguous as to whether the prepositional phrase *during the Triassic-Jurassic extinction event* modifies *gone* or *thought*. The correct interpretation is that the *during* prepositional phrase modify *gone*. However, the paraphrase of the correct parse, (3b), is not very fluent because the long prepositional phrase separates the verb and its complement, which causes the AMT workers to all choose (3c) as the best paraphrase. A disfluent paraphrase usually happens when the realizer needs to go far down the n-best realization list to find a realization which is different from the original sentence.

(3) a. They are *thought* to have *gone* extinct **during** the Triassic-Jurassic extinction event.
b. (gone→during): They are thought to have **gone during** the Triassic-Jurassic extinction event extinct.
c. (thought→during): They are **thought during** the Triassic-Jurassic extinction event to have gone extinct.

In other cases, although one parse is better than the other one for the disputed dependency, the rest of both parses is so broken that the realization cannot represent the meaning effectively. In those cases, the AMT workers usually could not give reliable annotations, because the realizations of the mangled parses make it hard for the AMT workers to see any reliable meaningful difference.

In some rare cases (3 out of 43), the AMT workers fail to choose the correct parse because they do not have the specific context to correctly understand the original sentence:

(4) a. Michigan's backup center, Gerald Ford, *expressed* a desire to *attend* the fair **while** in Chicago.
b. (attend→while): Michigan's backup center, Gerald Ford, expressed a desire to **attend while** in Chicago the fair.
c. (expressed→while): Michigan's backup center, Gerald Ford, **expressed while** in Chicago a desire to attend the fair.

The original sentence in (4a) is ambiguous as to whether the *while* adverbial phrase is modifying *attend* or *expressed*. After consulting the context of the Wikipedia article we know that when Gerald Ford made this speech, he was actually in Michigan and expressed this desire to visit the fair in Chicago. Accordingly, we annotated that *while* modifies *attend*. However, this information might not be available for the AMT workers. Also, perhaps because (4b) is a less fluent sentence where the *while* adverbial occurs between the verb *attend* and its object *the fair*, AMT workers all chose (4c) as the better paraphrase.

5.2 Regression analysis

We also conducted a regression analysis to determine the factors that affect AMT workers' choices. The predictors included in the analysis are ranks of the underlying parse of the paraphrase (parse), an arithmetic-mean approximation of BLEU between the paraphrase and the original sentence (bleu), and the fluency score of the paraphrase calculated by OpenCCG realizer, normalized globally across all the realizations in the data set (rlz.glb). For the two-sided paraphrases, all four predictors are calculated as the corresponding value of the paraphrase of the top parse minus the value of the paraphrase of the 'next' parse. The dependent variable in two-sided cases is 1 if the top parse is correct, 0 otherwise.

We fit four regression models respectively for the four combinations of majority (Maj) and strong majority (S. Maj) choices with one- and two-sided paraphrases. The regression analysis shows *bleu* has a significant effect on AMT workers' choice across all four settings. The positive coefficients of the predictor *bleu* indicates that AMT workers tend to choose the paraphrase that is similar to the original sentence in terms of its surface form. In some cases this likely means that the

	One-sided		Two-sided	
	Maj	S. Maj	Maj	S. Maj
parse	-0.03	-0.05	0.01	0.01
bleu	3.05*	4.38**	1.68*	3.07**
rlz.glb	0.01	0.01	0.07**	0.103***

Table 4: Coefficients of regression analysis of AMT workers' choice (significance codes are *: $p \leq 0.05$; **: $p \leq 0.01$; ***: $p \leq 0.001$)

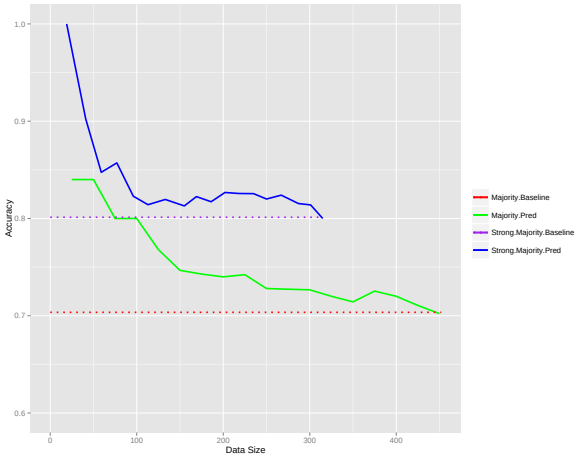

Figure 2: Accuracy and coverage trade-off plot for majority and strong majority choices

annotator is overly influenced by superficial similarity, which may partially explain the poor performance of one-sided paraphrases. We also observe a significant effect from the fluency score of the paraphrases in the two-sided case.

Inspired by the results above, we investigate the possibility of increasing the accuracy at the expense of coverage. We trained a logistic regression model on AMT workers' majority correct choices and plotted the accuracy of their choices in decreasing order of their likelihood of correctness, also plotting the accuracy of corresponding strong majority choices for comparison.

Figure 2 shows that in order to improve the accuracy of majority choices to 80%, we will lose around 80% data. However, the accuracy of strong majority choices, with 40% less coverage, is above 80%. Thus the results show that if we are willing to sacrifice some data coverage for higher quality annotation, strong majority choices are the better option. If data is quite plentiful (or nearly unlimited), only the most fluent items could be selected for annotation, in which case accuracy could potentially be pushed up past 90%.

	Dinosaur	Football
Train size	471	356
Eval size	291	226
Original acc.	0.701	0.668
Retrained acc.	**0.749**	**0.717**
Correction rate	0.243	0.32

Table 5: Parser retraining

6 Parser Retraining

As a preliminary experiment just using the validation data gathered to date, we retrained the OpenCCG parser with the majority judgments collected from AMT (along with the original CCGbank data). Results appear in Table 5. The training set of the dinosaur domain contains 471 parses and that of the football domain contains 356 parses, corresponding to the parses chosen by majority judgments of the AMT workers. We trained the OpenCCG parser on the two domains separately with ten-fold cross validation, and evaluated the parsing performance of the retrained parsers against our manually annotated gold dependencies (excluding 'neither' cases). Parses were considered correct if the parse matching the gold correct dependencies ranked higher than the parse matching the gold incorrect dependencies in the n-best list. For some sentences, we could not find a parse to match the annotated correct or wrong dependencies in the n-best list, especially the annotated wrong dependencies; we also excluded these sentences from the evaluation. In the end, we had 291 sentences in the dinosaur domain and 226 sentences in football for evaluation. *Original acc.* is the accuracy of the original OpenCCG parser evaluated on the gold annotated dependencies, while *Retrained acc.* is the accuracy of the retrained parsers and *Correction rate* is the proportion of original mistakenly parsed sentences that are correctly parsed by the retrained parsers.

MacNemar's chi-square test shows that the retrained parser achieves significantly higher accuracy in the dinosaur domain ($p = 0.02$). The same test on football data shows a trend but not a significant improvement ($p = 0.1$), most likely due to the smaller size of the training and evaluation sets for this domain. Meanwhile, the performance of the retrained parsers on the CCGbank development section does not differ significantly from the original parser ($p > 0.05$ for both).

7 Discussion

By directly asking AMT annotators to specify the attachment site for a PP, Jha et al. (2010) achieve 84% accuracy overall, rising to an impressive 95% in strong majority cases. However, their results are not directly comparable to our PP-disambiguation items since the texts are different and since they consider all PPs, rather than just the ones that the parser finds the most difficult. In addition, they allow annotators to indicate additional attachment sites if none of the automatically suggested ones are correct, yielding a considerably more complex annotation task than ours that requires explicit upfront instruction on the notion of PP-attachments; moreover, to extend their method to additional kinds of structural ambiguities, the instructions would be elaborated in each case.

The results and analysis indicate that the accuracy of our method could be improved simply by leaving aside the one-sided cases, where the AMT annotators may have been overly influenced by superficial similarity, as well as the passive rewrites, which performed much worse than the cleft rewrites on PP-attachment cases for reasons that are not clear. Realization fluency was also found to be a significant predictor of annotator choices in the two-sided cases, suggesting that accuracy could be further improved by taking this factor into account when selecting sentences if domain data is plentiful. Another alternative worth pursuing in future work would be to split sentences whose realizations are not sufficiently fluent, borrowing methods employed in syntactic simplification (Siddharthan, 2006; Siddharthan, 2011).

In future work we also plan to experiment with multiple parsers and additional collected data in order to measure the extent to which parsing performance on all attachments can be improved in new domains. Here we plan to use not only the OpenCCG reimplementation of the Hockenmaier & Steedman generative model, but also the Berkeley latent variable model and the Clark & Curran CCG parser, along with additional dependency parsers. To do so, we will take into account the "silver standard" nature of the annotations, namely that the parse corresponding to the selected disambiguating paraphrase may not be entirely correct, just closer than its competitor parse. In particular, using just the dependencies that differ between these two parses, we will select the highest-ranked parse that retains more of the correct (unlabeled, unordered) dependencies than any other in the n-best list. In this way, the dependencies yielded by each parser need not closely match the ones used to collect the data.

8 Conclusion

In this paper, we have shown that it is possible to obtain accurate crowd-sourced judgments of meaning by simply asking naive annotators to answer clarification questions, namely which of two automatically generated disambiguating paraphrases is closer to the original sentence in meaning. In a validation experiment, accuracy reached 80% or higher when there was a strong majority among the AMT annotators, both when using LF rewrites for PP-attachment and coordination ambiguities, as well as for direct reverse realizations, which cover a broader range of ambiguity types. Moreover, accuracy remains reasonably high for the subset of sentences where the top parse is incorrect, sufficiently so to enable a retrained parser to achieve significantly higher accuracy in a novel domain, even using just the limited validation data gathered to date. Data from the validation experiment is made available as a supplement to the paper.[7]

An analysis of errors revealed that one-sided cases (where only one disambiguating paraphrase could be generated) performed poorly, as did passive rewrites, and a regression analysis also revealed that realization fluency was a significant factor in predicting annotator decisions. In future work, we plan to take these lessons into account when collecting a much larger dataset in order to enable experiments on parser adaptation with multiple parsers, treating the crowd-sourced annotations as "silver standard" when retraining the parsers on in-domain sentences.

Acknowledgments

We thank James Curran, Eric Fosler-Lussier, the OSU Clippers Group and the anonymous reviewers for helpful comments and discussion. This work was supported in part by NSF grant 1319318.

[7] http://www.ling.osu.edu/~mwhite/data/law-x-2016-duan-hill-white-data.zip

References

[Boxwell and White2008] Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*.

[Clark and Curran2007] Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

[Deng et al.2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[Dredze et al.2007] Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1051–1055, Prague, Czech Republic, June. Association for Computational Linguistics.

[Duan and White2014] Manjuan Duan and Michael White. 2014. That's not what I meant! Using parsers to avoid structural ambiguities in generated text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 413–423, Baltimore, Maryland, June. Association for Computational Linguistics.

[Espinosa et al.2008] Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June. Association for Computational Linguistics.

[Fowler and Penn2010] Timothy A. D. Fowler and Gerald Penn. 2010. Accurate context-free parsing with Combinatory Categorial Grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344, Uppsala, Sweden, July. Association for Computational Linguistics.

[Gerdes2013] Kim Gerdes. 2013. Collaborative dependency annotation. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 88–97, Prague, Czech Republic, August. Charles University in Prague, Matfyzpress.

[Hockenmaier and Steedman2002] Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proc. ACL-02*.

[Hockenmaier and Steedman2007] Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

[Honnibal et al.2009] Matthew Honnibal, Joel Nothman, and James R. Curran. 2009. Evaluating a statistical CCG parser on Wikipedia. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 38–41, Suntec, Singapore, August. Association for Computational Linguistics.

[Jha et al.2010] Mukund Jha, Jacob Andreas, Kapil Thadani, Sara Rosenthal, and Kathleen McKeown. 2010. Corpus creation for new genres: A crowdsourced approach to PP attachment. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 13–20, Los Angeles, June. Association for Computational Linguistics.

[McClosky and Charniak2008] David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of ACL-08: HLT, Short Papers*, pages 101–104, Columbus, Ohio, June. Association for Computational Linguistics.

[Neumann and van Noord1992] Günter Neumann and Gertjan van Noord. 1992. Self-monitoring with reversible grammars. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 700–706, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Palmer et al.2005] Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1).

[Petrov et al.2006] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

[Rajkumar and White2010] Rajakrishnan Rajkumar and Michael White. 2010. Designing agreement features for realization ranking. In *Coling 2010: Posters*, pages 1032–1040, Beijing, China, August. Coling 2010 Organizing Committee.

[Russakovsky et al.2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Siddharthan2006] A. Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language & Computation*, 4(1):77–109.

[Siddharthan2011] Advaith Siddharthan. 2011. Text simplification using typed dependencies: A comparision of the robustness of different generation strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 2–11, Nancy, France, September. Association for Computational Linguistics.

[Steedman2000] Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

[White and Rajkumar2009] Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.

[White and Rajkumar2012] Michael White and Rajakrishnan Rajkumar. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255, Jeju Island, Korea, July. Association for Computational Linguistics.

[White2006] Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language & Computation*, 4(1):39–75.

[Winograd1973] Terry Winograd. 1973. A procedural model of language understanding. In Roger Schank and Ken Colby, editors, *Computer Models of Thought and Language*, pages 152–186. W.H. Freeman. Reprinted in Grosz et al. (eds), Readings in Natural Language Processing. Los Altos CA: Morgan Kaufmann Publishers, 1986, pp.249-266.

[Zeldes2016] Amir Zeldes. 2016. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, pages 1–32.