Dependency Tree Annotation with Mechanical Turk

Stephen Tratz
CCDC Army Research Laboratory
Adelphi, Maryland 20783 USA
stephen.c.tratz.civ@mail.mil

Abstract

Crowdsourcing is frequently employed to quickly and inexpensively obtain valuable linguistic annotations but is rarely used for parsing, likely due to the perceived difficulty of the task and the limited training of the available workers. This paper presents what is, to the best of our knowledge, the first published use of Mechanical Turk (or similar platform) to crowdsource parse trees. We pay Turkers to construct unlabeled dependency trees for 500 English sentences using an interactive graphical dependency tree editor, collecting 10 annotations per sentence. Despite not requiring any training, several of the more prolific workers meet or exceed 90% attachment agreement with the Penn Treebank (PTB) portion of our data, and, furthermore, for 72% of these PTB sentences, at least one Turker produces a perfect parse. Thus, we find that, supported with a simple graphical interface, people with presumably no prior experience can achieve surprisingly high degrees of accuracy on this task. To facilitate research into aggregation techniques for complex crowdsourced annotations, we publicly release our annotated corpus.

1 Introduction

State-of-the-art parsing models, which are important components to countless natural language processing workflows, are trained using treebanks of manually annotated parse trees. Unfortunately, many languages do not have treebanks and even the treebanks that do exist possess significant limitations in terms of size, genre, style, topic coverage, and/or other dimensions. Even the venerable Penn Treebank (Marcus et al., 1993)—one of the largest and most widely used treebanks—contains examples from only a single news source. Expanding existing treebanks or creating new ones tends to be quite expensive; for instance, the Prague Dependency Treebank, with over a million syntactically linked words, cost approximately $600,000 (Böhmová et al., 2003). In this work, we explore the use of crowdsourcing both to mitigate this cost barrier and also because, perhaps more importantly, it serves as a proof-of-concept for the case in which only non-experts are available to produce the parse trees, which is likely to be the situation for most under-resourced languages. Despite the widespread use of crowdsourcing to collect linguistic annotations, there have been few efforts to apply crowdsourcing to parsing—a fact we believe is largely due to concerns about the complexity of the task, the training requirements for the workers, as well as the skillfulness, diligence, and consistency of workers on crowdsourcing platforms such as Mechanical Turk.

This paper presents what is, to the best of our knowledge, the first use of Mechanical Turk or similar platform to crowdsource dependency parse trees. We request 10 annotations for each of 500 trees (250 from the Penn Treebank (PTB) and 250 from Wikipedia) and find that, despite not requiring any form of training, several of the Turkers who annotate 50 or more PTB sentences achieve at least 90% attachment agreement with the dependency conversion reference trees. Furthermore, for 72% of the PTB sentences, at least one Turker produces a tree that fully matches the reference. Ultimately, these results establish a baseline for what people with presumably no prior training can achieve in performing this challenging task.

2 Approach

To collect dependency tree annotations, we use Mechanical Turk's *external question* HIT (Human Intelligence Task) functionality. HITs are the basic unit of work on Mechanical Turk; essentially, they are questions to be answered, with an associated monetary reward. In the case of *external question*

Proceedings of the First Workshop on Aggregating and Analysing Crowdsourced Annotations for NLP, pages 1–5

https://doi.org/10.18653/v1/P17

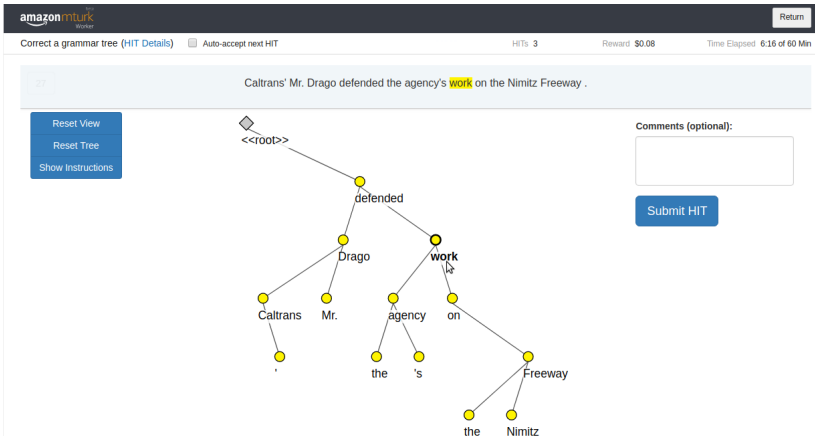

Figure 1: Screenshot of our annotation interface in the Mechanical Turk sandbox.

HITs, the annotation interface is hosted on an external website, which is embedded on the Mechanical Turk page within an HTML *iframe*. Each of our HITs involves constructing an unlabeled[1] dependency tree for a single sentence using the annotation interface described below. When Turkers submit their work, it goes both to the external server and to Amazon's Mechanical Turk website.

2.1 Graphical Annotation Interface

In our annotation interface (Tratz and Phan, 2018), shown in Figure 1, words are displayed as nodes and dependencies are displayed as edges between them.[2] Turkers create dependency arcs by dragging and dropping word nodes. Dropping one node onto another forms a dependency arc between the two, with the dragged node as the dependent of the latter. While dragging, green circular dropzones appear to highlight possible attachment sites.

The tool is configured to have all words initially attached to the dummy root node. The *Submit HIT* button becomes operable only when there is exactly one word connected to the dummy root. Thus, annotators are required to reattach all but one of the words.

Annotation guidelines are accessible by clicking the *Instructions* button, which brings up a box with the instructions that can be opened into a separate window. Since many Turkers may be reluctant to read wordy guides, our instructions consist primarily of 55 small example trees.

[1]We leave labeled dependency trees for future work, as labeling dependencies (e.g., 'subject', 'object') could be performed separately from the tree construction.

[2]Several aspects of the visual layout and styling of our tool are inspired by TrEd (Pajas and Štěpánek, 2008).

2.2 Data

For our Mechanical Turk HITs, we construct a dataset consisting of 250 sentences from the Penn Treebank (Marcus et al., 1993) and 250 from The Westbury Lab Wikipedia corpus (Shaoul, 2010), each consisting of 10 to 15 alphanumeric tokens. We ignore Penn Treebank sentences that merely report changes in earnings figures, prices, etc., since these types of sentences are particularly frequent in the Penn Treebank and make for an undiverse (and, therefore, uninteresting) sample. With the Wikipedia data, we filter incomplete and ungrammatical sentences,[3] and sentences with offensive language, heavy use of foreign words, or esoteric technical content. We merge various date expressions (e.g., *March 15, 2000*) into single tokens since these elements can be recognized with high accuracy using regular expressions.

2.3 HIT configuration parameters

We require that Turkers who work on our HITs reside in the USA or Canada, have a 95% or higher approval rating, and have previously had at least 20 other HITs approved. We pay $0.08[4] per completed assignment and request a total of 10 annotations per sentence (from 10 different workers).

2.4 Evaluation

For evaluation, we calculate both the percentage of words correctly attached (UAS: unlabeled attachment score) and the percentage of trees that

[3]Many of these errors may be due to the overly aggressive nature of our sentence splitter.

[4]In practice, this proved to be rather low for the amount of time Turkers appear to have been spending, so we gave out bonuses of $0.20 for most HIT assignments after all assignments were received.

Worker	Trees	Penn Treebank				Wikipedia			
		Trees	UAS	FTM	$\widetilde{\text{time}}$	Trees	UAS	FTM	$\widetilde{\text{time}}$
W1	177	90	0.921	0.500	53.5	87	0.935	0.552	51
W2	453	223	0.913	0.439	37	230	0.918	0.465	33
W3	499	249	0.906	0.454	44	250	0.907	0.428	41
W4	410	201	0.901	0.443	42	209	0.901	0.407	38
W5	412	194	0.840	0.211	40	218	0.865	0.261	36
W6	450	226	0.796	0.159	45.5	224	0.831	0.228	38.5
W7	411	207	0.774	0.077	54	204	0.792	0.127	47.5
W8	434	211	0.724	0.057	55	223	0.768	0.112	44
W9	119	61	0.724	0.115	34	58	0.708	0.138	35.5
W10	352	178	0.644	0.034	45.5	174	0.688	0.052	39
W11	197	107	0.500	0.000	111	90	0.518	0.000	94
W12	128	59	0.423	0.000	311	69	0.434	0.000	238
W13	379	185	0.228	0.000	48	194	0.245	0.000	46
A1	500	250	0.969	0.712	—	250	1.000	1.000	—

Table 1: Results for the 13 workers (W1–W13) who annotate 50 or more Penn Treebank trees, including the total number of trees annotated, unlabeled attachment scores (UAS), full tree match rate (FTM), and median time in seconds ($\widetilde{\text{time}}$) between accepting a HIT and submitting results. For reference, we also include scores for the primary author (A1).

fully match the reference (FTM: full tree match rate). In the case of the Penn Treebank sentences, the reference is the automatic dependency conversion; for the Wikipedia sentences, we use the primary author's annotation. A total of 112 Turkers participate; however, most only annotate 1 or 2 sentences, making it difficult to estimate their aptitude for this task. The 13 workers who annotated 50 or more sentence account for over 88% of the annotations received. The scores for these 13 most prolific annotators are presented in Table 1, along with the scores for the primary author (who is, ideally, representative of an expert annotator) included as well for comparison.

2.5 Results Discussion

We note a high degree of variation in the quality of the work of the different annotators. Four of the more prolific Turkers achieve attachment scores of 90% or higher on the Penn Treebank portion of the data, but others are unable to reach even 50% agreement. To examine how the Turkers' performance varies with time, we plot the change to their overall attachment scores as they perform annotations (see Figure 2). Several annotators, including *W6*, *W8*, *W10*, and *W11*, improve noticeably early on as they gain experience. A couple annotators (i.e., *W2* and *W7*) show slight decreases in their scores, which may be due to fatigue. Overall, there appears to be a very high degree of consistency over time for the individual Turkers.

For 72% of the Penn Treebank sentences, at least one annotator produces a dependency tree that fully matches the reference completely. Taking the Turker-provided tree that best matches the reference for each of the PTB sentences results in a set of trees with a 96.8% attachment score, 3257 of 3363 attachments agreeing. Examining the remaining 106 disagreements in more detail, we observe that approximately 13 are due to handling of business suffixes (e.g., Corp., Co.), at least 8 are due to errors in the reference, and 5 are related to quantifying adverbs preceding expressions with numbers (e.g., *about 8 %*). Many of these disagreements could be brought into alignment by tweaking the annotation guide and/or fixing bugs in the dependency conversion. The remaining errors fall into a wide variety of categories. A substantial portion are related to phenomena that are somewhat challenging to represent well with dependency parses, such as gapping, right node raising, and *it* extraposition.

In general, the results seem to suggest that the Wikipedia sentences may be slightly easier to parse, but, overall, the results are quite similar to those for the PTB sentences.

3 Related Work

To date, there have been very few efforts to crowdsource parsing and no efforts, to the best of our knowledge, to do so directly with a full parse tree editor like ours other than in small classroom studies like that of Gerdes (2013).

In what is the most closely related line of work, researchers build and deploy a Game with a Purpose (GWAP) (Von Ahn, 2006) called ZOM-

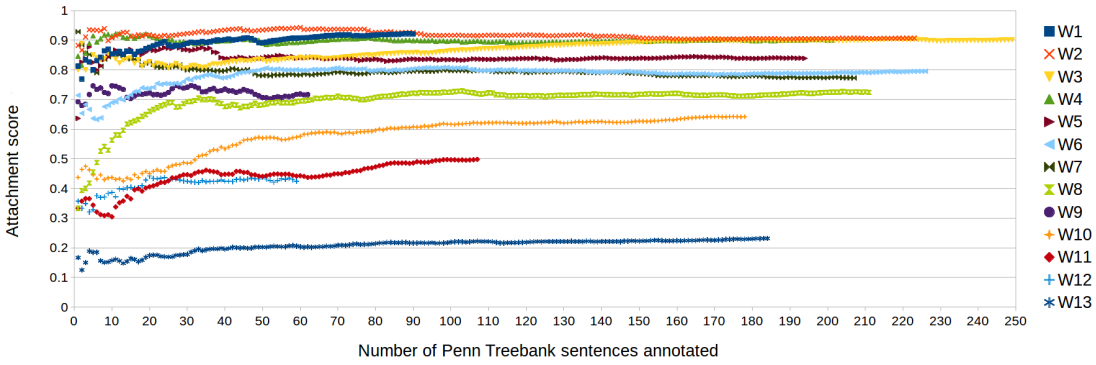

Figure 2: Overall unlabeled attachment score on Penn Treebank sentences for the 13 most prolific workers (W1–W13), calculated as each annotation is received. (Note: Workers do not annotate sentences in the same order.)

BILINGO in order to crowdsource a French treebank (Fort et al., 2014; Guillaume et al., 2016; Fort et al., 2017). ZOMBILINGO participants, who are all volunteers, work on one attachment decision at a time, using the metaphor that the governing word is devouring the child just as zombies seek out brains. The participants must complete a training phase for each dependency relation type that they work on, and they are unable to proceed to relations deemed more difficult until they have demonstrated some skill on less challenging dependency relations.

Another related effort is that of He et al. (2016), who, in an initial step toward human-in-the-loop parsing, crowdsource individual attachment annotations by asking annotators multiple choice questions automatically generated using parse trees produced by an existing parser. They are able to achieve some performance gains, including a 0.2 F1 improvement on their in-domain corpus (from 88.1 to 88.3) and a 0.6 F1 improvement on their out-of-domain corpus (from 82.2 to 82.8).

It is worth noting that there are a number of ethical concerns regarding the use of Mechanical Turk, and a variety of articles have been written on this subject. We refer the reader to a discussion of these issues by Fort et al. (2011).

4 Conclusion and Future Work

This paper details the first effort to crowdsource treebanking using Mechanical Turk or similar online crowdsourcing platform. Using our graphical web-based treebanking tool, we collect 10 dependency parse annotations for each of 500 sentences. Despite not requiring any training, several of the annotators achieve attachment scores at or above 90%. Although this may not be of sufficient quality to train a competitive English parser, it establishes a baseline for dependency tree annotation using workers who are presumably non-experts, demonstrating the potential value that non-experts can bring to parsing annotation projects. Moreover, treebanks with 90% attachment accuracy would still be useful for other languages, especially those with little or no annotated data. To this end, we plan to investigate whether our approach will result in comparable accuracy for other languages, which will likely require recruiting workers outside of Mechanical Turk.

We find that taking the best tree for each of the 250 Penn Treebank sentences results in a dataset that agrees with the Penn Treebank dependency conversion on 96.8% of attachments and agrees with the full dependency tree 72% of the time. Though unreasonable to assume that any such oracle exists, it may be possible to approach this level of accuracy by employing a multi-step approach in which workers review and judge the work of others, similar to the translation crowdsourcing efforts of Zaidan and Callison-Burch (2011). To facilitate research among the greater community into techniques for aggregating complex crowdsourced annotations, we provide our annotated dataset at `https://github.com/USArmyResearchLab/ARL_CrowdTree`.

Finally, we plan to integrate active learning algorithms that run while the Turkers are annotating. A *query by committee* (Seung et al., 1992) framework would be a natural choice—multiple different parsing models would learn from the submitted trees and the sentences provided to the annotators would be selected based upon the level of disagreement between the parsers.

References

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Springer Netherlands.

Karën Fort, Gilles Adda, and K Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 37(2):413–420.

Karën Fort, Bruno Guillaume, and Hadrien Chastant. 2014. Creating Zombilingo, a Game With A Purpose for dependency syntax annotation. In *Proceedings of the First International Workshop on Gamification for Information Retrieval*, pages 2–6.

Karën Fort, Bruno Guillaume, and Nicolas Lefebvre. 2017. Who wants to play Zombie? A survey of the players on ZOMBILINGO. In *Proceedings of Games4NLP: Using Games and Gamification for Natural Language Processing*.

Kim Gerdes. 2013. Collaborative Dependency Annotation. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 88–97.

Bruno Guillaume, Karen Fort, and Nicolas Lefebvre. 2016. Crowdsourcing Complex Language Resources: Playing to Annotate Dependency Syntax. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 3041–3052.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-Loop Parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2337–2342.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):330.

Petr Pajas and Jan Štěpánek. 2008. Recent Advances in a Feature-Rich Framework for Treebank Annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 673–680. Association for Computational Linguistics.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by Committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294.

Cyrus Shaoul. 2010. The Westbury Lab Wikipedia Corpus. *Edmonton, Alberta: University of Alberta*.

Stephen Tratz and Nhien Phan. 2018. A Web-based System for Crowd-in-the-Loop Dependency Treebanking. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, pages 2189–2193.

Luis Von Ahn. 2006. Games with a Purpose. *Computer*, 39(6):92–94.

Omar F Zaidan and Chris Callison-Burch. 2011. Crowdsourcing Translation: Professional Quality from Non-Professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1220–1229.