Human Computation and Crowdsourcing: Works in Progress and Demonstration Abstracts
An Adjunct to the Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing

Crowdsourcing in Language Classes Can Help Natural Language Processing

Barbora Hladká, Jirka Hana, Ivana Lukšová
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{hana,hladka,luksova} (at) ufal.mff.cuni.cz

Abstract

One way of teaching grammar, namely morphology and syntax, is to visualize sentences as diagrams capturing relationships between words. Similarly, such relationships are captured in a more complex way in treebanks serving as key building stones in modern natural language processing. However, building treebanks is very time consuming, thus we have been seeking for an alternative cheaper and faster way, like crowdsourcing. The purpose of our work is to explore possibility to get sentence diagrams produced by students and teachers. In our pilot study, the object language is Czech, where sentence diagrams are part of elementary school curriculum.

Natural Language Processing (NLP) deals with computer and human interaction mainly in written natural language. Modern NLP systems employ algorithms that are mostly based on supervised machine learning methods. These methods require data to train on, e.g., treebanks. A treebank is a collection of sentences manually annotated with some linguistic information. We highlight treebanks with morphological and syntactic information that serve as training data for an essential NLP procedure performing syntactic analysis, so called parser. It is impossible to list only one number to illustrate the state-of-the-art performance of parsers. However, the parser accuracy does not exceed 90 % mostly. For example, parsing web English texts achieves the accuracy in the 80-84 % range of F-measure (Petrov and McDonald 2012). Therefore, there is still room for improvement, e.g., through getting more training data. However, the annotation process is very resource consuming, thus we have been seeking for alternative ways of faster and cheaper annotation. Namely, we have been inspired by the solution of crowdsourcing, see e.g., (Brabham 2013).

Teaching grammar is an essential part of language classes. One way to teach morphology and syntax is to draw sentence diagrams (hence SDs) capturing relationships between words in the sentence. The discussion on teaching grammar through sentence diagrams is beyond the scope of this paper, for inspiration see `http://teach-grammar.com`. Studying diagrams from a perspective of data for parsing, they could be of a great importance.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The main purpose of our work is to explore the possibility of *getting sentence diagrams* produced by students and their teachers and *using* them as training data for parsers. Since sentence diagrams and treebank annotation schemes usually differ, specific transformation rules have to be specified. For Czech, we introduced these rules in (Hana and Hladká 2012). In this paper, we focus on getting diagrams only. We established two language-independent goals to reach and consider together: (i) design a tool for drawing SDs that attract teachers to use it in language classes and encourage students to use it for practicing on their own; (ii) ensure that the quality and quantity of the obtained data satisfy requirements for applying supervised learning methods. The goals have been established in general without a reflection of actual practice in both teaching grammar and treebanking for a language under consideration.

Čapek editor Traditionally, diagrams are only in students notebooks so they are not accessible to us at all. Since we require diagrams electronically, we have been developing a sentence diagram editor *Capek*. We design it both as a CALL (Computer-Assisted Language Learning) system for practicing morphology and syntax and as a crowdsourcing system for getting data. The editor can be used for drawing sentence diagrams in any natural language. The editor will manipulate three different sets of sentences:

- *Sentences* – raw sentences
- *Analyzed sentences* – sentences with a set of SDs. For each SD, there will be available the number of votes assigned by teachers and students. A teacher/student will vote for a given SD if (s)he likes either a whole SD or some edges.
- *Agreed sentences* – sentences with only one SD. A criterion for selecting it: some function of #s (student votes) and #t teacher votes.

The editor will run in three modes:

- *Student* – (interactive) practicing home (drawing SD from scratch, checking SDs from *Analyzed*). Help available on demand if analysing sentences from *Agreed sentences*.
- *Teacher* – drawing SDs from sctratch, checking SDs from *Analyzed*, preparing exercises.

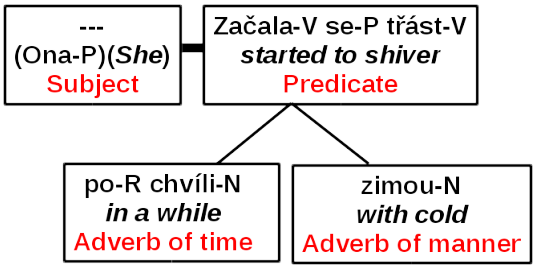

Figure 1: A sample of sentence diagram
(Noun, Pronoun, pReprosition, Verb)

- *Class* – interactive practicing, discussing

Both students and teachers will be awarded points for any activity.

Data quality Data quality belongs to the most important issues related to crowdsourcing, see e.g., (Hsueh, Melville, and Sindhwani 2009). We discuss the data quality from two aspects:

1. evaluation of teachers'/students' diagrams against other teachers'/students' diagrams, i.e., we consider how diagrams are similar;
2. combination of diagrams of one sentence to get a better diagram, i.e., we deal with multiple, possibly noisy, annotations and we study if they are useful.

Pilot study

In our pilot study, the object language is Czech, where sentence diagrams are part of elementary school curriculum.

Sentence diagrams In the Czech sentence diagrams, a sentence is represented as a directed acyclic graph (roughly a tree) with labeled words and nodes. The words are labeled with part-of-speech tags. The nodes correspond to single words mostly. However, multiple nodes (e.g., for a preposition with its noun, or complex predicates) and empty nodes (for dropped subjects) are also possible. The edges capture the dependency relation between nodes (e.g., between an object and its predicate). Node labels expresse the type of dependency, or syntactic function. For illustration, let's consider the sentence in (1) and its diagram in Figure 1:

(1) (—) Začala se po chvíli třást zimou.
She started in a while to shiver with cold.
'She started to shiver with cold in a while.'

Capek 1.0 The editor is designed as a client-server application. It exists as a desktop application, written in Java on top of the Netbeans Platform, `http://platform.netbeans.org` and as a web application `http://capek.herokuapp.com/`. The current version runs in the *Student* mode and enables users to select a sentence from *Analyzed*, do both morphological and syntactic analysis, and send the analyses to the server.

Data We randomly selected a workbench of 101 sentences from a Czech language textbook for elementary schools. Initially, these sentences were manually analyzed by elementary school teachers T1 and T2 and secondary school students S1 and S2 using Čapek. The set of *Analyzed* was initialized with the 101 sentences having up to 4 different SDs.

Data quality More details on this issue are provided in (Hana, Hladká, and Lukšová 2014).

We compute the *similarity between sentence diagrams* using a tree edit distance, inspired by (Bille 2005). It assumes two sentence diagrams (source and target) and three edit operations: relabeling a node, deleting a non-root node, and inserting a node. A source SD is transformed into a target SD by a sequence of edit operations. Each operation has a particular cost, the cost of the sequence is simply the sum of the cost of individual operations. Then *tree edit distance* between two sentence diagrams is the cost of a cheapest sequence of operations turning one diagram into another.

To *combine multiple diagrams*, we have used a majority-voting method: each candidate node and edge is assigned a score based on the number of votes it received from the input diagrams. The votes for edges are weighted by a specific criterion. To build a final diagram, we first create its set of nodes, then its set of edges linking final nodes, and finally extend the set of nodes by any empty nodes. The method can produce both nodes and edges that do not occur in any of the input diagrams.

Initial Experiments As we expected, the teachers' diagrams are the most similar ones and on the other hand, the students' diagrams are the most different one.

A group of 7 graduate and undergraduate students drew diagrams for 10 sentences randomly selected from the 101 workbench using Čapek 1.0. We merged their analyses and compared them to the diagrams by the T1 teacher.

References

Bille, P. 2005. A survey on tree edit distance and related problems. *Theoretical computer science* 337(1):217–239.

Brabham, D. C. 2013. *Crowdsourcing*. MIT Press.

Hana, J., and Hladká, B. 2012. Getting more data: Schoolkids as annotators. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, 4049–4054.

Hana, J.; Hladká, B.; and Lukšová, I. 2014. Sentence diagrams: their evaluation and combination. In *Proceedings of the 8th Linguistic Annotation Workshop (LAW VIII 2014)*.

Hsueh, P.-Y.; Melville, P.; and Sindhwani, V. 2009. Data quality from crowdsourcing: A study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, 27–35.

Petrov, S., and McDonald, R. 2012. Overview of the 2012 Shared Task on Parsing the Web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).