A Visualized Toolkit for Crowdsourcing NLP Annotations

Hanchuan Li, Haichen Shen, Shengliang Xu and Congle Zhang
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{hanchuan,haichen,shengliang,clzhang}@cs.washington.edu

1 Introduction

With the popularity of the Intenet, there are huge amount of text in the web, and their size is still growing quickly. In order to use them, it is of utmost importance to develop automatic nature language processing (NLP) systems to handle the big data.

It has been wildly accepted that statistical machine learning approaches are very effective for most NLP problems, such as parsing (Klein and Manning, 2003), information extractions (Banko et al., 2007), question answering (Kwok et al., 2001) and so on. However, there is a significant limitation of all the statistical approaches: they need a lot of training data to learn models. Training data are labeled by human annotators. In most cases, annotators manually predict the desired outputs for the inputs. The algorithms then learn from the training data the statistics and models, which are used to automatically predict the output of the unlabeled data.

Generally, dataset labeling for machine learning problems (often classification) is time consuming and tedious. To make matters worse, it is even harder for annotators to label NLP problems than to label standard classification problems.

For a standard classification problem, human annotators are given a set of labels and a list of objects. Their jobs are to choose a label for each object. Since objects are usually independent of each other, local information is enough for label prediction. Besides, labeling errors, if any, would not affect the rest of the dataset. Therefore, the user interface for the annotations could be very simple: plain text or Excel tables are often enough in many cases.

But for NLP annotations, the outputs are often structured predictions. That is, each prediction depends on some other prediction. For example, Figure 1 shows how a parsing algorithm converts a sentence into a tree. It is hard for annotators to decide the position of a single word in the tree before drawing the whole tree. Besides, a large group of NLP problems are related to clustering, such as coreference resolution (Lee et al., 2011), which clusters two or more expressions in a text refer to the same person or thing. Unlike classification, annotates must acquire some global information before correctly labeling the clusters. Suppose there are "Jeffery Heer", "Jeff Bilmes", "Jeff", "the professor" in a text. To decide whether which "Jeff" and which "professor", annotator must read the context to know who they are. Transitivity also makes the clustering annotation very tricky. For example, after merging some pairs of points, the annotator has created two clusters {"Jeff, Jeff Bilmes"} and {"Jeffery Heer, Professor Heer"}. He then accidentally merges "Jeff" and "Jeffery Heer". It would immediately result in a very large and incorrect cluster.

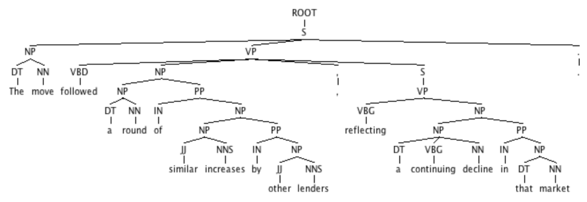

Figure 1: An example parse tree.

The challenges listed above make the labeling process uncomfortable for normal annotators. Firstly, annotators must spend a lot of time to understand the global structure of the data before labeling anything. Secondly, annotators would often revisit and edit their labels. In the above example, the annotator notices that “Jeff Bilmes” and “Jeffery Heer” are in the same cluster, then he has to go back and fix that error. Such trial and errors could cause conflicts and incomplete annotations. In fact, even trained linguistic must spend a lot of time to label NLP datasets. A famous example is that it costs 8 years to create the Penn Tree Bank, a labeled dataset of parsed trees.

Nowadays, there are many worker at crowdsourcing platforms like Mechanical Turks[1] and Odesk. They provide us opportunities to quickly collect a large amount of training data. But most workers are normal people, having little knowledge about linguistic, and having no reason to be patient enough. If the problem is to hard, they would switch to other easier and profitable jobs. So it is impractical to ask them to label over plain text files or Excel tables for NLP annotations.

2 Proposed System

In this project, we aim to develop a visualized toolkit for crowd-sourcing NLP annotations. The target audience are normal people with little knowledge and patience. The toolkit would allow them to quickly label NLP datasets.

There are two key properties of our toolkit: firstly, annotators could interact with the data to understand them in a refresh way. Annotators label some examples and they expect immediate feedback from the toolkit. These feedbacks will help them understand the problem. Secondly, the toolkit should enable and encourage trial and errors. It would not assume any edits from the users as gold, but treat the edits as clues to better visualize the data to the annotator. When the annotator finishes a labeling task, he should be satisfied and confident with the overall outcome. For example, it is hard to distinguish whether “Jeff” is “Jeff Bilmes” or “Jeffery Heer” when data points are seen individually. But if the toolkit could immediate show a big cluster {“Jeff, Jeff Bilmes, Jeffery Heer, Professor Heer”} after incorrectly merge two points, the annotators would have a good chance to fix it.

In this project, we would focus on two important kinds of NLP annotations: building trees (*e.g.* parsing) and clustering (*e.g.* coreference resolution). But we would keep in mind that the toolkit should be easily extensible to any NLP problems.

[1] `https://www.mturk.com/mturk/welcome`

2.1 Building tree

Formally, the input is a list of nodes, the output would be a tag for each node. Nodes sharing the same tag stay in the same cluster. Let us use the following running example: the task is to co-refer five mentions “Jeffery Heer”, “Jeff Bilmes”, “Jeff”, “Professor Heer” and “Mr Bilmes”. We propose to use “dependency wheel” for this task. At first, annotator would see 5 objects listed on the right side of the window, and there is nothing on the wheel. He could operate the data in the following ways:

Drag node a to node b when they are in the same cluster: suppose we drag “Jeff” to “Jeffery Heer”. They would be added to the wheel if not yet. Suppose “Jeff”, “Jeffery Heer” are in clusters {“Jeff, Jeff Bilmes”} and {“Jeffery Heer, Professor Heer”}. Then the resulting cluster would be {“Jeff, Jeff Bilmes, Jeffery Heer, Professor Heer”}. All four nodes will be colored the same and group together on the wheel. There is also an “yes” edge between “Jeff” and “Jeffery Heer” highlighting this operator.

Two nodes are different: annotators soon notice that “Jeff Bilmes” and “Jeffery Heer” in the resulting cluster is bad. But he is not aware of which previous edit causes this error. So he just tell the toolkit that “Jeff Bilmes” and “Jeffery Heer” are two different entities. There comes a “no” edge between them. The system would figure out this different edge is conflict with the previous “yes” edge between “Jeff” and “Jeffery Heer”. The toolkit would highlight the conflicts so annotator could cancel one or several of them.

Tag the nodes: nodes with the same tag would be grouped together and put together on the wheel. The tag could cause conflicts. The toolkit would automatically figure them out and highlight them on the screen. So annotators could easily trial and errors.

2.2 Tree Generation

Formally, the input is a list of nodes, the output would be a tree whose leaf nodes are the input nodes. Besides, breadth-first search the tree will not change the initial order of the nodes. Let us parse the sentence “My little dog also likes eating sausage.” as a running example. At first, annotators would see 6 individual nodes. He could operate the tree in the following ways:

Drag node A to node B when they are siblings: if two nodes have no parent node yet, we would create an inner node as the parent of the two children, and add tree edges. For example, user can drag "little" to "dog" to create a node for phrase "little dog". If B has its parent C already, A will be linked to C as its new child. For example, user can drag "My" to "little" or "dog" to merge "my" with "little dog". This operator is enough to build the tree.

Click the edge to cancel parent-child relationship: when the parent node loses all its children, the inner node will be removed from the tree. This operator enables trial and error for annotators.

Tag the nodes: when linguistics build the parse tree, they would also tag the POS tags (*i.e.* verb, noun phrase). Since there are dozens tags on the tree, it is too complicated for normal people to tag all of them. But fortunately, a partial set of tags, especially verb and nouns, would help the machine learning algorithms a lot. Annotators could choose to tag some nodes with those most important tags. We would also support edit and delete the tags.

3 Evaluation methods

We are going to do separate evaluations of the two targeted labeling tasks.

- Tree building. There are a lot of tree builiding tasks in the NLP area. We propose to evaluate our system using the task of semantic tree parsing of sentences. The goal of the task is to get the semantic parsing tree of each labeling sentence.

  Participants: We propose to gather 4 to 10 participants from undergraduate/graduate UW students.

  Evaluation process: Each participant is given 2 sets of randomly selected sentences. In each set, there may be around 100 sentences. All the participants are divided into two groups. Half of the participants will be asked to complete the parsing of the first given sentence set using only plain text file editors, plain html files or Microsoft Excel files without diagrams. They are then required to complete the second setence set with the aid of our tool. The other half of the participants will be asked to complete the labeling tasks in the reverse order. The purpose of the two group splitting and the different order of conducting the labeling is to eliminate the possible factor of gaining unexpected information from using the first labeling tool.

  Evaluation: Both time consumption and result quality of the two groups of participants will be evaluated.

- Clustering building. In order to evaluate the clustering building task, we propose to use entity clustering in NLP relation extraction. The goal of the task is to get a cluster structure of given entities.

  Participants: We propose to also gather 4 to 10 participants from UW undergraduate/graduate students.

  Evaluation process: Each participant is given 2 sets of randomly selected entities, e.g. human names, together with extra context info for entity resolution. The participants are also divided into two groups as in the tree building task. And also the two groups label the entities in the same way as in tree building task.

  Evaluation: Both time consumption and result quality of the two groups of participants will be evaluated.

4 Related work

Manual annotation for NLP training data is well-known for its tedium and large amount of data. To generate a comprehensive annotated training set requires much human effort. Annotators are also prone to make mistakes during the long and tedious annotating process. Researchers are trying to address these problems by two means: 1) develop visualization tools to improve annotation efficiency as well as reduce the error rate in annotation; 2) adopt crowdsourcing to enable collaborative annotation that accelerates the process of annotation.

Most related in scope is (Yan and Webster, 2012) which provides a collaborative tool to assist annotators in tagging of complex Chinese and multilingual linguistic data. It visualizes a tree model that represents the complex relations across different linguistic elements to reduce the learning curve. Besides it proposes a web-based collaborative annotation approach to meet the large amount of data. Their tool focuses on a specific area —- complex multilingual linguistic data, whereas our work is trying to address how to generate a visualization model for general data sets.

Most related in scope is (Yan and Webster, 2012) which provides a collaborative tool to assist annotators in tagging of complex Chinese and multilingual linguistic data. It visualizes a tree model that represents the complex relations across different linguistic elements to reduce the learning curve. Besides it proposes a web-based collaborative annotation approach to meet the large amount of data. Their tool only focuses on a specific area that is complex multilingual linguistic data, whereas our work is trying to address how to generate a visualization model for general data sets.

Crowdsourcing now is recognized as a growing and promising approach in NLP. Many related works focus on conceptual study and formalization of crowdsourcing. For instance, (Quinn and Bederson, 2009) categorizes crowdsourcing into seven genres: Mechanized Labor, Game with a Purpose (GWAP), Widom of Crowds, Crowdsourcing, Dual-Purpose Work, Grand Serarch, Human-based Genetic Algorithms and Knowledge Collection from Volunteer Contributors. Other works, such as (Abekawa et al., 2010) and (Irvine and Klementiev, 2010), develops a specific tool and verifies the feasibility and benefit of crowdsourcing. Nevertheless, we seek to provide an intuitive visualization to lower the barrier to get started on crowdsourcing.

References

Takeshi Abekawa, Masao Utiyama, Eiichiro Sumita, and Kyo Kageura. 2010. Community-based construction of draft and final translation corpus through a translation hosting site minna no hon'yaku (mnh). In *LREC*. Citeseer.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676.

Ann Irvine and Alexandre Klementiev. 2010. Using mechanical turk to annotate lexicons for less commonly used languages. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 108–113. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Cody Kwok, Oren Etzioni, and Daniel S Weld. 2001. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. Association for Computational Linguistics.

Alexander J Quinn and Benjamin B Bederson. 2009. A taxonomy of distributed human computation. *Human-Computer Interaction Lab Tech Report, University of Maryland*.

Hengbin Yan and Jonathan Webster. 2012. Collaborative annotation and visualization of functional and discourse structures. In *Proceedings of the Twenty-Fourth Conference on Computational Linguistics and Speech Processing*, pages 366–374.