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 the size of the texts are still growing with enormous speed. In order to use them, it is of utmost importance to develop automatic nature language processing (NLP) algorithms to handle tbe large amount of data.

A typical NLP system would first employ syntactic processing and then employ semantic processing. Syntactic processing is often task independent. It aims to convert the raw text into some machine friendly structures. The pipeline often includes tokenization, POS tagging, parsing, name entity recognition and coreference. Semantic processing is often task dependent. It tries to exploit useful information from the text for an end task.

It has been wildly accepted that statistical machine learning approaches are the most effectively approach for most NLP problems. Statical NLP approaches have shown their powers on syntactic parsing, relation extractions, question answering and so on. However, there is a signification limitation of all these statistical approach, it requires large amounts of training data to learn the system. Training data are labeled by users. Such annotation process asks annotators to manually generate the output for the data. Statistical NLP algorithms learn the statistics and models from these training data and then use them to predict the unlabeled data automatically.

Generally speaking, labeling datasets 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. They are asked to choose a label for each object. Objects are usually independent of each other, so each prediction is often straightforward. Annotators could make some errors, but these errors would not affect the rest of the dataset. The user interface for annotations could be very naive, most people use EXCEL tables for this purpose.

But for NLP annotations, the predictions are often structured predictions. That is, individual outputs are not independent, but are related to each other. For example, Figure 1 shows how parsing algorithm converts a sentence into a tree. Usually, annotates cannot decide the position of a single word in the tree before drawing the whole tree. Another challenge is that, a large group of NLP problems are related to clustering, such as coreference which clusters mentions in the article of the same entities. Unlike classification, annotates must understand the big picture in order to correctly label the clusters of the data. Transitivity makes the annotation very tricky. For example, after merging pairs of points, the annotator has created two clusters $\{A_1, \ldots, A_{10}\}$ and $\{B_1, \ldots, B_{10}\}$. He then accidentally merges A_1 and B_2. If this operation is incorrect, it would immediately cause 100 pairs of errors, which is really a disaster.

The challenges listed above makes the labeling process extremely uncomfortable for normal annotators. Firstly, annotators must spend a lot of time to understand the inner structure of the data before labeling anything. Secondly, annotators would often revisit and edit their labels. For example, during coreference annotation, at first annotators did not know there are two "Obama"s in the article so he simply annotate all "Obama" as "Barack Obama". After a while, he have to go back to fix them be-

cause he noticed "Mitchell Obama". False annotations tend to occur during such trial and errors. In fact, even trained linguistic must spend a lot of time to label NLP datasets. For example, it costs 8 years to create the Penn Tree Bank, a labeled set of parsed trees.

Nowadays, there are many people work at crowdsourcing platforms like Mechanical Turks and Odesk. This phenomena provides an opportunity to quickly collect a large amount of training data. But most of them are normal people, having little knowledge about linguistic, and having no reason to be patient enough. It is impractical to ask them to label over plain text files or Excel tables, since they would switch to other easier profitable tasks.

2 Proposed System

In this project, we aim to develop a visualized toolkit for crowdsourcing 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 points to understand the data in an up-to-date way. For example, any partial annotations reflect annotators' partial understanding. So they would expect immediate feedback from the toolkit. Secondly, the toolkit should enable and even encourage trial and errors. It would not take any edits from the users as granted, but treat the edits as clues to better render the data to the annotator. When the annotator finish a labeling task, he should be satisfied with the global outcome and confident. In the above $\{A_1, \ldots, A_{10}\}$ and $\{B_1, \ldots, B_{10}\}$ example, it is likely that A_1 and B_2 are hard to distinguish when the pair is seen separately from the rest of the data. But if the toolkit could immediate show a big cluster $\{A_1, \ldots, A_{10}, B_1, \ldots, B_{10}\}$ after accidentally merge A_1 with B_2, the annotators would have a good chance to change their mind and fix the errors.

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

2.1 Annotate Tree Prediction

2.2 Annotate Graph Prediction