We present a system for improving the accuracy of one NLP technique, Named Entity Recognition or NER, on Twitter data by training a recognizer specifically for this type of data. This training data is obtained from the Amazon Mechanical Turk crowdsourcing platform.

Our experiments were conducted on data collected by a third party between March 2007 and November 2008. It consists of approximately 150 million status updates from 1.5 million users, which are stored in a database and indexed with Lucene.

Amazon Mechanical Turk (MTurk) is a system which allows people (“requesters”) who need simple tasks done by humans to find other people (“workers”) to do those tasks. We used this system to generate training data for our machine learning algorithm which will accomplish the end goal.

Using Mechanical Turk is as simple as defining Human Intelligence Tasks (“HITs”) which are basically a template for the webpage that workers will eventually interact with, and providing a spreadsheet with a list of values that should be fill this template. Our HIT (filled in with example data) looks like this:

We had a total of 270 workers, who completed 3,080 HITs. Each HIT was composed of five tweets: one which had been previously annotated by us (“Gold”), and four which had not been annotated. Each HIT was given to two different workers, in order to build better consensus.

Annotating our data with MTurk was cheap, but there are difficulties resulting from the informal nature of the annotators and potential for disagreement between them. There is also the consideration that unethical workers may use robots to fill out the form, resulting in completely invalid answers. To address this problem, we developed an algorithm based on Google’s PageRank which determines inter-worker agreement. Pseudocode is shown below.

We can take advantage of these conventions to do a better job recognizing words that may refer to entities. For example, hash tags are often used when people tweet from events, mentioning the event by a hashtag.

Twitter is a platform for users to post short messages (“status updates” or “statuses”). These messages are often informal, leading to poor grammar, punctuation, and capitalization. These special features lead to poor performance from traditional NER systems. Other features of the Twitter platform that set it apart from other types of text include the conventions that have accumulated around it:

- Hashhtags are words or abbreviations prefixed with a “#” character, which are used either as metadata or as a word.
- User mentions consist of a username preceded by an @ character, and can be used to name a person or direct a tweet at them.
- Retweets are designated by the characters “RT”, followed by a username and the text of a tweet that user posted.
- URLs are also often present in tweets, using the familiar http:// notation, but often use an external URL shorter like bit.ly or is.gd to fit better within the 140 character limit of Twitter.

We take advantage of these conventions to do a better job recognizing words that may refer to entities. For example, hash tags are often used when people tweet from events, mentioning the event by a hashtag.

Results

Amazon Mechanical Turk proved an effective and inexpensive method of gathering training data for our machine learning algorithm. Using a purpose-designed algorithm, we can take raw data from MTurk and produce high-quality training data in a reasonable amount of time.

Future Work

In the future we hope to train NER systems on many chosen subsets of the AMT data, in order to judge the impact that a single exceptionally good or bad worker can have on accuracy of the trained NER system. With this data, we hope to compare the similarity metric we have established to the individual worker’s efficacy, in order to formally establish its usefulness as a worker quality metric.

Open Questions

- Could an online version (i.e., one that takes new incremental data over time) of the MTurk agreement algorithm allow for real-time spammer detection? This would allow better use of time, since when a spammer is detected they can be disallowed from completing more HITs, letting other legitimate workers do the HITs instead, so that fewer HITs need to be resubmitted.
- How many annotations are necessary to train a good NER system? How can we detect this condition with a machine learning algorithm?
- If we can build a system that automatically trains itself: Could a NER system bootstrap itself via AMT? The system could take a large arbitrary corpus and automatically submit jobs to AMT, then train itself based on those results.