Semi-Automated Named Entity Annotation

Kuzman Ganchev and **Fernando Pereira**
Computer and Information Science,
University of Pennsylvania,
Philadelphia PA
{ kuzman and pereira } @cis.upenn.edu

Mark Mandel
Linguistic Data Consortium,
University of Pennsylvania, Philadelphia PA
mamandel@ldc.upenn.edu

Steven Carroll and **Peter White**
Division of Oncology, Children's Hospital of Philadelphia Philadelphia PA
{ carroll and white }@genome.chop.edu

Abstract

We investigate a way to partially automate corpus annotation for named entity recognition, by requiring only binary decisions from an annotator. Our approach is based on a linear sequence model trained using a k-best MIRA learning algorithm. We ask an annotator to decide whether each mention produced by a high recall tagger is a true mention or a false positive. We conclude that our approach can reduce the effort of extending a seed training corpus by up to 58%.

1 Introduction

Semi-automated text annotation has been the subject of several previous studies. Typically, a human annotator corrects the output of an automatic system.

The idea behind our approach is to start annotation manually and to partially automate the process in the later stages. We assume that some data has already been manually tagged and use it to train a tagger specifically for high recall. We then run this tagger on the rest of our corpus and ask an annotator to filter the list of suggested gene names.

The rest of this paper is organized as follows. Section 2 describes the model and learning algorithm. Section 3 relates our approach to previous work. Section 4 describes our experiments and Section 5 concludes the paper.

2 Methods

Throughout this work, we use a linear sequence model. This class of models includes popular tagging models for named entities such as conditional random fields, maximum entropy Markov models and max-margin Markov networks. Linear sequence models score possible tag sequences for a given input as the dot product between a learned weight vector and a feature vector derived from the input and proposed tas sequence. Linear sequence models differ principally on how the weight vector is learned. Our experiments use the MIRA algorithm (Crammer et al., 2006; McDonald et al., 2005) to learn the weight vector.

2.1 Notation

In what follows, x denotes the generic input sentence, $Y(x)$ the set of possible labelings of x, and $Y^+(x)$ the set of correct labelings of x. There is also a distinguished "gold" labeling $y(x) \in Y^+(x)$. For each pair of a sentence x and labeling $y \in Y(x)$, we compute a vector-valued feature representation $f(x, y)$. Given a weight vector w, the score $w \cdot f(x, y)$ ranks possible labelings of x, and we denote by $Y_{k,w}(x)$ the set of k top scoring labelings for x.

We use the standard B,I,O encoding for named entities (Ramshaw and Marcus, 1995). Thus $Y(x)$ for x of length n is the set of all sequences of length n matching the regular expression $(\texttt{O}|(\texttt{BI}*))*$. In a linear sequence model, for suitable feature functions f, $Y_{k,w}(x)$ can be computed efficiently with Viterbi decoding.

2.2 k-best MIRA and Loss Functions

The learning portion of our method finds a weight vector w that scores the correct labelings of the test data higher than incorrect labelings. We used a k-

Proceedings of the Linguistic Annotation Workshop, pages 53–56,
Prague, June 2007.

best version of the MIRA algorithm (Crammer et al., 2006; McDonald et al., 2005). This is an online learning algorithm that starts with a zero weight vector and for each training sentence makes the smallest possible update that would score the correct label higher than the old top k labels. That is, for each training sentence x we update the weight vector w according to the rule:

$$\begin{aligned} & w_{\text{new}} = \arg\min_w \|w - w_{\text{old}}\| \\ & \text{s. t. } w \cdot f(x, y(x)) - w \cdot f(x, y) \geq L(Y^+(x), y) \\ & \qquad \forall y \in Y_{k, w_{\text{old}}}(x) \end{aligned}$$

where $L(Y^+(x), y)$ is the *loss*, which measures the errors in labeling y relative to the set of correct labelings $Y^+(x)$.

An advantage of the MIRA algorithm (over many other learning algorithms such as conditional random fields) is that it allows the use of arbitrary loss functions. For our experiments, the loss of a labeling is a weighted combination of the number of false positive mentions and the number of false negative mentions in that labeling.

2.3 Semi-Automated Tagging

For our semi-automated annotation experiments, we imagine the following scenario: We have already annotated half of our training corpus and want to annotate the remaining half. The goal is to save annotator effort by using a semi-automated approach instead of annotating the rest entirely manually.

In particular we investigate the following method: train a high-recall named entity tagger on the annotated data and use that to tag the remaining corpus. Now ask a human annotator to filter the resulting mentions. The mentions rejected by the annotator are simply dropped from the annotation, leaving the remaining mentions.

3 Relation to Previous Work

This section relates our approach to previous work on semi-automated approaches. First we discuss how semi-automated annotation is different from active learning and then discuss some previous semi-automated annotation work.

3.1 Semi-Automated versus Active Learning

It is important not to confuse semi-automated annotation with active learning. While they both attempt to alleviate the burden of creating an annotated corpus, they do so in a completely orthogonal manner. Active learning tries to select which instances should be labeled in order to make the most impact on learning. Semi-automated annotation tries to make the annotation of each instance faster or easier. In particular, it is possible to combine active learning and semi-automated annotation by using an active learning method to select which sentences to label and then using a semi-automated labeling method.

3.2 Previous work on semi-automated annotation

The most common approach to semi-automatic annotation is to automatically tag an instance and then ask an annotator to correct the results. We restrict our discussion to this paradigm due to space constraints. Marcus et al. (1994), Chiou et al. (2001) and Xue et al. (2002) apply this approach with some minor modifications to part of speech tagging and phrase structure parsing. The automatic system of Marcus et al. only produces partial parses that are then assembled by the annotators, while Chiou et al. modified their automatic parser specifically for use in annotation. Chou et al. (2006) use this tag and correct approach to create a corpus of predicate argument structures in the biomedical domain. Culota et al. (2006) use a refinement of the tag and correct approach to extract addressbook information from e-mail messages. They modify the system's best guess as the user makes corrections, resulting in less annotation actions.

4 Experiments

We now evaluate to what extent our semi-automated annotation framework can be useful, and how much effort it requires. For both questions we compare semi-automatic to fully manual annotation. In our first set of experiments, we measured the usefulness of semi-automatically annotated corpora for training a gene mention tagger. In the second set of experiments, we measured the annotation effort for gene mentions with the standard fully manual method and with the semi-automated methods.

4.1 Measuring Effectiveness

The experiments in this section use the training data from the the Biocreative II competition (Tanabe et

Sentence	Expression of SREBP-1a stimulated StAR promoter activity in the context of COS-1 cells
gold label	Expression of [SREBP-1a] stimulated [StAR promoter] activity in ...
alternative	Expression of [SREBP-1a stimulated StAR promoter] activity in ...
alternative	Expression of [SREBP-1a] stimulated [StAR] promoter activity in ...

Figure 1: An example sentence and its annotation in Biocreative II. The evaluation metric would give full credit for guessing one of the alternative labels rather than the "gold" label.

al., 2005). The data is supplied as a set of sentences chosen randomly from MEDLINE and annotated for gene mentions.

Each sentence in the corpus is provided as a list of "gold" gene mentions as well as a set of alternatives for each mention. The alternatives are generated by the annotators and count as true positives. Figure 1 shows an example sentence with its gold and alternative mentions. The evaluation metric for these experiments is F-score augmented with the possibility of alternatives (Yeh et al., 2005).

We used 5992 sentences as the data that has already been annotated manually (set **Data-1**), and simulated different ways of annotating the remaining 5982 sentences (set **Data-2**). We compare the quality of annotation by testing taggers trained using these corpora on a 1493 sentence test set.

We trained a high-recall tagger (recall of 89.6%) on **Data-1**, and ran it on **Data-2**. Since we have labels available for **Data-2**, we simulated an annotator filtering these proposed mentions by accepting them only if they exactly match a "gold" or alternative mention. This gave us an F-score of 94.7% on **Data-2** and required 9981 binary decisions.

Figure 2 shows F_1 score as a function of the number of extra sentences annotated. Without any additional data, the F-measure of the tagger is 81.0%. The two curves correspond to annotation with and without alternatives. The horizontal line at 82.8% shows the level achieved by the semi-automatic method (when using all of **Data-2**).

From the figure, we can see that to get comparable performance to the semi-automatic approach, we need to fully manually annotate roughly a third as much data with alternatives, or about two thirds as much data without alternatives. The following section examines what this means in terms of annotator time by providing timing results for semi-automatic and fully-manual annotation without alternatives.

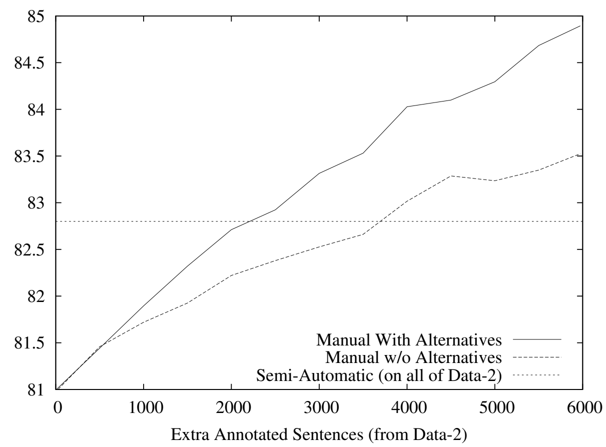

Figure 2: Effect of the number of annotated instances on F_1 score. In all cases the original 5992 instances were used; the curves show manual annotation while the level line is the semi-automatic method. The curves are averages over 3 trials.

4.2 Measuring Effort

The second set of experiments compares annotator effort between fully manual and semi-automatic annotation. Because we did not have access to an experienced annotator from the Biocreative project, and gene mention annotations vary subtly among annotation efforts, we evaluated annotator effort on on the PennBioIE named entity corpus.[1] Furthermore, we have not yet annotated enough data locally to perform both effectiveness and effort experiments on the local corpus alone. However, both corpora annotate gene mentions in MEDLINE abstracts, so we expect that the timing results will not be significantly different.

We asked an experienced annotator to tag 194 MEDLINE abstracts: 96 manually and 98 using the semi-automated method. Manual annotation was done using annotation software familiar to the annotator. Semi-automatic annotation was done with a

[1]Available from `http://bioie.ldc.upenn.edu/`

Web-based tool developed for the task. The new tool highlights potential gene mentions in the text and allows the annotator to filter them with a mouse click. The annotator had been involved in the creation of the local manually annotated corpus, and had a lot of experience annotating named entities. The abstracts for annotation were selected randomly so that they did not contain any abstracts tagged earlier. Therefore, we did not expect the annotator to have seen any of them before the experiment.

To generate potential gene mentions for the semi-automated annotation, we ran two taggers on the data: a high recall tagger trained on the local corpus and a high recall tagger trained on the Biocreative corpus. At decode time, we took the gene mentions from the top two predictions of each of these taggers whenever there were any gene mentions predicted. As a result, the annotator had to make more binary decisions per sentence than they would have for either training corpus alone. For the semi-automated annotation, the annotator had to examine 682 sentences and took on average 10 seconds per sentence. For the fully-manual annotation, they examined 667 sentences and took 40 seconds per sentence on average. We did not ask the annotator to tag alternatives because they did not have any experience with tagging alternatives and we do not have a tool that makes the annotation of alternatives easy. Consequently, effort totals for annotation with alternatives would have been skewed in our favor. The four-fold speedup should be compared to the lower curve in Figure 2.

5 Discussion and Further Work

We can use the effort results to estimate the relative effort of annotating without alternatives and of semi-automated annotation. To obtain the same improvement in F-score, we need to semi-automatically annotate roughly a factor of 1.67 more data than using the fully manual approach. Multiplying that by the 0.25 factor reduction in annotation time, we get that the time required for a comparable improvement in F-score is 0.42 times as long – a 58% reduction in annotator time.

We do not have any experiments on annotating alternatives, but the main difference between semi-automated and fully-manual annotation is that the former does not require the annotator to decide on boundaries. Consequently, we expect that annotation with alternatives will be considerably more expensive than without alternatives, since more boundaries have to be outlined.

In future work, it would be interesting to compare this approach to the traditional approach of manually correcting output of a system. Due to constraints on annotator time, it was not possible to do these experiments as part of the current work.

References

Fu-Dong Chiou, David Chiang, and Martha Palmer. 2001. Facilitating treebank annotation using a statistical parser. In *HLT '01*. ACL.

Wen-Chi Chou, Richard Tzong-Han Tsai, Ying-Shan Su, Wei Ku, Ting-Yi Sung, and Wen-Lian Hsu. 2006. A semi-automatic method for annotating a biomedical proposition bank. In *FLAC'06*. ACL.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7.

Aron Culota, Trausti Kristjansson, Andrew McCallum, and Paul Viola. 2006. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence*, 170:1101–1122.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL'05*. ACL.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*. ACL.

Lorraine Tanabe, Natalie Xie, Lynne H. Thom, Wayne Matten, and W. John Wilbur. 2005. GENETAG: a tagged corpus for gene/protein named entity recognition. *BMC Bioinformatics*, 6(Suppl. 1).

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of the 19th international conference on Computational linguistics*. ACL.

Alexander Yeh, Alexander Morgan, Marc Colosimo, and Lynette Hirschman. 2005. BioCreAtIvE Task 1A: gene mention finding evaluation . *BMC Bioinformatics*, 6(Suppl. 1).