Penn/Umass/CHOP Biocreative II systems

Kuzman Ganchev [1] kuzman@seas.upenn.edu
Koby Crammer [1] crammer@seas.upenn.edu
Fernando Pereira [1] pereira@seas.upenn.edu
Gideon Mann [2] gmann@cs.umass.edu
Kedar Bellare [2] kedarb@cs.umass.edu
Andrew McCallum [2] mccallum@cs.umass.edu
Steve Carroll [3] carroll@genome.chop.edu
Yang Jin [3] jin@genome.chop.edu
Peter White [3] white@genome.chop.edu

[1] Department of Computer and Information Science, University of Pennsylvania, Philadelphia PA
[2] Department of Computer Science, University of Massachusetts, Amherst MA
[3] Division of Oncology, The Children's Hospital of Philadelphia, Philadelphia PA

Abstract

Our team participated in the entity tagging and normalization tasks of Biocreative II. For the entity tagging task, we used a k-best MIRA learning algorithm with lexicons and automatically derived word clusters. MIRA accommodates different training loss functions, which allowed us to exploit gene alternatives in training. We also performed a greedy search over feature templates and the development data, achieving a final F-measure of 86.28%. For the normalization task, we proposed a new specialized on-line learning algorithm and applied it for filtering out false positives from a high recall list of candidates. For normalization we received an F-measure of 69.8%.

Keywords: aberrant splicing, database, point mutation, scanning model

1 Introduction

For the entity tagging task, we used a rule-based tokenizer followed by a linear sequence model trained using a 5-best MIRA learning algorithm. MIRA accommodates different training loss functions, which allowed us to exploit alternative taggings in training and tune the loss function for higher performance. We further augment the feature set with curated lexicons and with automatically derived unsupervised word clustering features and find that their combination gives a more than additive gain.

For the normalization task, we used our entity tagging system trained for high-recall and use this to generate a list of mentions for each abstract. We used a simple matching algorithm to find potential gene aliases for each mention, and a new specialized online learning algorithm to filter out false positives from this initial high-recall set of candidates. Some of the features used by the learning algorithm are based on learning what kinds of changes indicate different aliases for the same gene and what kinds indicate different genes.

2 Entity Tagging

Training and test sentences are first tokenized with a rule-based tokenizer. A linear sequence model assigns one of the three `B`,`I`, and `O` tags to each word, as in [13]. We start from a CRF-based system [4] with features tuned by greedy search (Section 2.4). We then make four major changes:

1. Training via k-best MIRA on alternative labelings (Section 2.1).
2. Tuning the MIRA loss function (Section 2.1).

3. Including curated lexicon features (Section 2.2)

4. Adding unsupervised clustering word features (Section 2.3).

Together, these changes yielded an overall improvement of 5% absolute performance improvement (28% relative error reduction) over the baseline system.

2.1 *k*-best MIRA and Loss Functions

In what follows, x denotes the generic input sentence, $Y(x)$ denotes the set of possible labelings of x, $Y^+(x)$ 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. As with hidden Markov models [12], for suitable feature functions f, $Y_{k,w}(x)$ can be computed eficiently by dynamic programming. A linear sequence model is given by a weight vector w.

The learning portion of our method requires finding a weight vector w that scores the correct labelings of the test data higher than incorrect labelings. We used a k-best version of the MIRA algorithm [2, 8, 7]. This is an online learning algorithm that for each training sentence x updates the weight vector w according to the rule:

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

where $L(Y^+(x), y)$ is a measure of the loss of labeling y with respect to the set of correct labelings $Y^+(x)$. We experimented with different loss functions, and finally settled on a loss that is a weighted combination of the number of false positive gene mentions and false negative gene mentions in the sentence. Our experience showed that getting a high precision is relatively easier than a high recall, so we weigh the number of false negatives higher than the number of false positives.

2.2 Lexicons

In our experiments, we used a number of curated lexicons. We used the lexicons as proposed in [10]. During the greedy feature search we found that removing some of these actually improved performance as did adding others. We added a list of genes developed at CHOP, a set of names of chemicals extracted from PubChem and a list of disease terms extracted from the MeSH ontology.

2.3 Clustering

An 85 million word subset of MEDLINE was used to cluster words by bigram language model perplexity into a binary tree [1]. Different depth tree cuts were then applied to produce 5 clustering features at differet levels of granularity for each word type in the tree [11]. Thus, for each word type that has been clustered there are 5 different non-independent cluster features generated by the clustering. These additional features are then added the feature function in training and testing. On our development data, adding these features produced a 0.7% improvement in the best system and as much as 1.3% improvement in inferior systems.

2.4 Greedy Search

For the baseline system, we started from a feature set similar to that of [10]. We proceeded to improve it as follows.

Method	CRF			Mira Hamming			Mira FP+2FN		
Features	P	R	F-1	P	R	F-1	P	R	F-1
Public access	83.4	79.1	81.2	85.5	78.3	*81.8*	83.0	86.3	*84.6*
Clusters	84.6	80.6	*82.6*	85.7	79.0	82.2	83.1	86.9	85.0
Lexicons	85.0	80.1	*82.5*	86.7	80.1	83.3	84.4	86.9	85.7
All Resources	**85.9**	**81.3**	**83.5**	**87.4**	**81.6**	**84.4**	**85.1**	**87.7**	**86.4**

Table 1: F-measure on held out testing data (not used in development). Our best system applied four techniques (lexicons, clustering, alternative labelings in training, and tuned loss) for a 5% absolute improvement (28% error reduction) over the baseline system.

We divide our features into sets of feature templates. For example, there are many features for the identity of the current token (one for each token type), but we group all of these into a single feature template. Starting with our initial list of feature templates, we repeatedly remove the one whose removal results in the greatest increase in the score of the development data, until no further improvement is possible. Removing just one feature template in this way requires training one model for each removal candidate. Once we can't improve the development data performance, we start adding feature templates from a list of candidates. This resulted in some unexpected additions and non-additions. For example, we found that adding a conjunction of four POS tags helps performance, while adding our list of gene acronyms actually hurts performance.

Even though there are hundreds of thousands of features, there are only dozens of feature templates, so doing this optimization on the development data doesn't lead to very severe overfitting: the F-score of the final system on the development data was within 1% of that on unseen data. The initial performance of the baseline system is similar to that of the "public access" system described in the following section. This is despite the fact that the baseline system had access to some lexicons, due to a poorer selection of features overall.

2.5 Analysis

While we were developing the system, we split the provided data into a training, development and test set (80%,10% and 10% respectively). We did not use the testing data during development, so we can use it to compare a few approaches. This comparison is shown in Table 1. The table shows Precision, Recall and F-score for a conditional random field, MIRA with the standard loss and MIRA with a loss of the number of false positives plus the number of false negatives times two.[1] The features used for the experiments were the final features after the search over feature templates described in Section 2.4 without lexicon features and cluster features (public access), with the lexicons but no cluster features, with the cluster features but no lexicons and with lexicons and cluster features.

Table 1 demonstrates that out of the three improvements, the use of MIRA with a tuned loss function and using the alternative labelings yielded the highest performance improvement over the baseline CRF system (3%). Use of alternative labelings alone yielded only around .5% performance improvement. Both the automatic cluster features and the curated lexicons gave around 1% F-measure improvement over the baseline. In conjunction with MIRA with the tuned loss function, the lexicons still provide around a 1% improvement over the baseline MIRA system, while the the automatic clusters provide only around a .5% improvement. Surprisingly, in the final system, adding both lexicons and clusters yielded a 1.8% improvement.

[1]We found that high recall is harder to achieve than high precision, so we weigh false negatives more.

3 Gene Normalization

Our approach is in some ways similar to that of []*ryanBC1B. Like them, we first generate a high-recall list of candidate mentions with corresponding aliases and gene IDs, and then use a linear classifier to filter out the false positives. Our system differs in the way that we find an intial list of candidates (Section 3.1,) the learning algorithm (Section 3.3) and in the features that we used to make the decisions (Section 3.2).

3.1 Gene Mentions and Matching

We used the gene tagger described in Section 2. Since our approach for the gene mention task will filter out incorrect gene mentions later, we used a loss function to maximize recall: the loss of a labeling for a sentence was set to the number of false negatives (with respect to the true labeling). This resulted in a recall on the gene mention tagging task of a little over 90%.[2]

For each gene mention in the high-recall list, we return the list of gene ID/alias pairs where the alias is the same as the text of the mention after normalization steps that include removal of common words (such as "gene", "protein", "mouse"), replacement of digits with the corresponding roman numerals, removal of spaces and dashes, and case conversion. This yields an initial candidate list that has a recall of 77.2% but a precision of only 37.3%. We tried more agressive matching rules, but we found that this makes the next learning stage too difficult for the small amount of available training data.

3.2 Filtering Features

To avoid overfitting due to the small training set, we used only 89 features, including the number of candidates competing for the same mention, the number of candidates that agree on the gene ID, and which of "human", "rat" and "mouse" appears closest to the mention. A set of more complicated but very useful features are based on a learned string string alignment model [6].

The string edit distance model is trained to maximize the probability alignments between aliases of the same gene, while minimizing the probability of alignments between aliases of different genes. For each candidate, the model gives a probability that the gene mention to the alias, which is converted into the following features: the binned value of the probability, the rank of the current candidate among all candidates in the abstract, and the rank of the current candidate among candidates for the same mention. These features resulted in a significant improvement (about 2% in F-measure) in our cross-validation development runs.

3.3 Learning Algorithm

Our model learns to distinguish a candidate (mention-alias pair) containing a true mention of a gene from a false positive Unfortunately, the training data is incomplete: for each abstract we have only a list of gene IDs. To overcome this problem we created a MIRA-inspired (Section 2.1) online learning algorithm that makes use of the correct gene IDs (given to us), as well as its current predictions to figure out which candidates to require be correct.

More formally, let C be the set of candidates for a particular abstract, C_{top} be the set of current highest scoring candidates for each correct gene ID, and C_{FP} be the set of candidates that correspond to an incorrect gene ID, but were scored above some threshold θ by the current model. Then, our algorithm performs the following update

$$\begin{array}{rcll} w_{\text{new}} & = & \arg\min_w \|w - w_{old}\| + \gamma \max_c \xi_c & \\ & s.t. & w \cdot c \geq 1 - \xi_c & \forall c \in C_{\text{top}} \\ & & -w \cdot c \geq 1 - \xi_c & \forall c \in C_{\text{FP}} \\ & & \xi_c \geq 0 & \forall c \in C_{\text{FP}} \cup C_{\text{top}}. \end{array}$$

[2] We cannot estimate this exactly because we used all available labeled data to train the model.

Here the ξ_c are slack variables that allow us to sometimes fail to separate the data. For our experiments, we used a θ of 0.2 and a γ in the range $0.005 \leq \gamma \leq 0.5$.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by DoD contract #HM1582-06-1-2013. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsor.

References

[1] Peter Brown, Peter deSouza, Robert Mercer, Vincent Della Pietra, and Jenifer Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[2] Koby Crammer. *Online Learning of Complex Categorial Problems*. PhD thesis, Hebrew Univeristy of Jerusalem, 2004.

[3] Kuzman Ganchev, Fernando Pereira, Gideon Mann, Andrew McCallum, Steve Carroll, Yang Jin, and Peter White. Online learning, greedy search and unsupervised clustering for gene mention tagging. Submission to BioCreative II, 2006.

[4] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, 2001.

[5] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[6] Andrew McCallum, Kedar Bellare, and Fernando Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. In *Conference on Uncertainty in AI (UAI)*, 2005.

[7] Ryan McDonald, Koby Crammer, and Fernando Pereira. Flexible text segmentation with structured multilabel classification. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 987–994, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[8] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[9] Ryan McDonald, Jay Crim, and Fernando Pereira. Automatically annotating documents with normalized gene lists. In *BMC Bioinformatics*, 2005.

[10] Ryan McDonald and Fernando Pereira. Identifying gene and protein mentions in text using conditional random fields. In *BMC Bioinformatics*, 2005.

[11] Scott Miller, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 337–342, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[12] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):275–285, 1989.

[13] Lance Ramshaw and Mitch Marcus. Text chunking using transformation-based learning. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey, 1995. Association for Computational Linguistics.

[14] Erik Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-200 shared task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nédellec, and Erik Tjong Kim Sang, editors, *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, pages 127–132. Association for Computational Linguistics, Somerset, New Jersey, 2000.