Weakly Supervised Learning Methods for Improving the Quality of Gene Name Normalization Data

Ben Wellner
wellner@mitre.org

The MITRE Corporation
202 Burlington Rd
Bedford MA 01730

Computer Science Department
Brandeis University
Waltham MA 02454

Abstract

A pervasive problem facing many biomedical text mining applications is that of correctly associating mentions of entities in the literature with corresponding concepts in a database or ontology. Attempts to build systems for automating this process have shown promise as demonstrated by the recent BioCreAtIvE Task 1B evaluation. A significant obstacle to improved performance for this task, however, is a lack of high quality training data. In this work, we explore methods for improving the quality of (noisy) Task 1B training data using variants of weakly supervised learning methods. We present positive results demonstrating that these methods result in an improvement in training data quality as measured by improved system performance over the same system using the originally labeled data.

1 Introduction

A primary set of tasks facing biomedical text processing systems is that of categorizing, identifying and classifying entities within the literature. A key step in this process involves grouping mentions of entities together into equivalence classes that denote some underlying entity. In the biomedical domain, however, we are fortunate to have structured data resources such as databases and ontologies with entries denoting these equivalence classes. In biomedical text mining, then, this process involves associating mentions of entities with known, *existing* unique identifiers for those entities in databases or ontologies – a process referred to as *normalization*. This ability is required for text processing systems to associate descriptions of concepts in free text with a grounded, organized system of knowledge more readily amenable to machine processing.

The recent BioCreAtIvE Task 1B evaluation challenged a number of systems to identify genes associated with abstracts for three different organisms: mouse, fly and yeast. The participants were provided with a large set of noisy training data and a smaller set of higher quality development test data. They were also provided with a lexicon containing all the potential gene identifiers that might occur and a list of known, though incomplete, names and synonyms that refer to each of them.

To prepare the training data, the list of unique gene identifiers associated with each full text article was obtained from the appropriate model organism database. However, the list had to be pruned to correspond to the genes mentioned in the abstract. This was done by searching the abstract for each gene on the list or its synonyms, using exact string matching. This process has the potential to miss genes that were referred to in the abstract using a phrase that does not appear in the synonym list. Additionally, the list may be incomplete, because not all genes mentioned in the article were curated, so there are mentions of genes in an abstract that did not have a corresponding identifier on the gene list.

This paper explores a series of methods for attempting to recover some of these missing gene

Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics, pages 1–8, Detroit, June 2005.

identifiers from the Task 1B training data abstracts. We start with a robust, machine learning-based baseline system: a reimplementation of the system in [1]. Briefly, this system utilizes a classifier to select or filter matches made against the synonym list with a loose matching criterion. From this baseline, we explore various methods for re-labeling the noisy training data, resulting in improved scores on the overall Task 1B development test and evaluation data. Our methods are based on weakly supervised learning techniques such as co-training [2] and self-training [3, 4] for learning with both labeled and unlabeled data.

The setting here is different than the typical setting for weakly supervised learning, however, in that we have a large amount of *noisily* labeled data, as opposed to completely *un*labeled data. The main contribution of this work is a framework for applying weakly supervised methods to this problem of re-labeling noisy training data.

Our approach is based on partitioning the training data into two sets and viewing the problem as two mutually supporting weakly supervised learning problems. Experimental results demonstrate that these methods, carefully tuned, improve performance for the gene name normalization task over those previously reported using machine learning-based techniques.

2 Background and Related Work

2.1 Gene Name Normalization and Extraction

The task of normalizing and identifying biological entities, genes in particular, has received considerable attention in the biological text mining community. The recent Task 1B from BioCreAtIvE [5] challenged systems to identify unique gene identifiers associated with paper abstracts from the literature for three organisms: mouse, fly and yeast. Task 1A from the same workshop focused on identifying (i.e. tagging) mentions of genes in biomedical journal abstracts.

2.2 NLP with Noisy and Un-labeled Training Data

Within biomedical text processing, a number of approaches for both identification and normalization of entities have attempted to make use of the many available structured biological resources to "bootstrap" systems by deriving noisy training data for the task at hand. A novel method for using noisy (or "weakly labeled") training data from biological databases to learn to identify relations in biomedical texts is presented in [6]. Noisy training data was created in [7] to identify gene name mentions in text. Similarly, [8] employed essentially the same approach using the FlyBase database to identify normalized genes within articles.

2.3 Weakly Supervised Learning

Weakly supervised learning remains an active area of research in machine learning. Such methods are very appealing: they offer a way for a learning system provided with only a small amount of labeled training data and a large amount of un-labeled data to perform better than using the labeled data alone. In certain situations (see [2]) the improvement can be substantial.

Situations with small amounts of labeled data and large amounts of unlabeled data are very common in real-world applications where labeling large quantities of data is prohibitively expensive. Weakly supervised learning approaches can be broken down into *multi*-view and *single*-view methods.

Multi-view methods [2] incrementally label unlabeled data as follows. Two classifiers are trained on the training data with different "views" of the data. The different views are realized by splitting the set of features in such a way that the features for one classifier are conditionally independent of features for the other *given the class label*. Each classifier then selects the most confidently classified instances from the unlabeled data (or some random subset thereof) and adds them to the training set. The process is repeated until all data has been labeled or some other stopping criterion is met. The intuition behind the approach is that since the two classifiers have different views of the data, a new training instance that was classified with high confidence by one classifier (and thus is "redundant" from that classifier's point of view) will serve as an informative, novel, new training instance for the other classifier and vice-versa.

Single-view methods avoid the problem of finding an appropriate feature split which is not possible or appropriate in many domains. One common approach here [4] involves learning an ensemble of

classifiers using *bagging*. With bagging, the training data is randomly sampled, with replacement, with a separate classifier trained on each sample. Un-labeled instances are then labeled if *all* of the separate classifiers agree on the label for that instance. Other approaches are based on the expectation maximization algorithm (EM) [9].

3 System Description

The baseline version of our system is essentially a reproduction of the system described in [1] with a few modifications. The great appeal of this system is that, being machine learning based, it has no organism-specific aspects hard-coded in; moving to a new organism involves only re-training (assuming there is training data) and setting one or two parameters using a held-out data set or cross-validation.

The system is given a set of abstracts (and associated gene identifiers at training time) and a lexicon. The system first proposes candidate phrases based on all possible phrases up to 8 words in length with some constraints based on part-of-speech[1]. Matches against the lexicon are then carried out by performing exact matching but ignoring case and removing punctuation from the both the lexical entries and candidate mentions. Only *maximal* matching strings were used – i.e. sub-strings of matching strings that match the same id are removed.

The resulting set of matches of candidate mentions with their matched identifiers results in a set of *instances*. These instances are then provided with a label - "yes" or "no" depending on whether the match in the abstract is correct (i.e. if the gene identifier associated with the match was annotated with the abstract). These instances are used to train a binary maximum entropy classifier that ultimately decides if a match is valid or not.

Maximum entropy classifiers model the conditional probability of a class, **y**, (in our setting, **y**="yes" or **y**="no") given some observed data, **x**. The conditional probability has the following form in the binary case (where it is equivalent to logistic regression):

$$P(y \mid x) = \frac{\exp(\sum_i \lambda_i f_i(x, y))}{Z(x)}$$ where Z(x) is the normalization function, the λ_i are real-valued model parameters and the f_i are arbitrary real-valued feature functions.

One advantage of maximum entropy classifiers is the freedom to use large numbers of statistically non-independent features. We used a number of different feature types in the classifier:

- the matching phrase
- the matched gene identifier
- the previous and subsequent two words of the phrase
- the number of words in the matching phrase
- the total number of genes that matched against the phrase
- all character prefixes and suffixes up to length 4 for words within the phrase

An example is shown below in Figure 1 below.

Abstract Excerpt:

"This new receptor, **TOR** (thymus orphan receptor)…"

Feature Class	Specific Feature
Phrase	TOR
GENEID	MGI104856
Previous-1	,
Previous-2	receptor
Subsequent-1	(
Subsequent-2	thymus
Number of Matches	2
Number of Words	1
Prefix-1	T
Prefix-2	TO
Prefix-3	TOR
Suffix-1	R
Suffix-2	OR
Suffix-3	TOR

Figure 1. An abstract excerpt with the matching phrase "TOR". The resulting features for the match are detailed in the table.

In addition to these features we created additional features constituting conjunctions of some of these "atomic" features. For example, the conjoined feature **Phrase=TOR AND GENEID=MGI104856** is "on" when both conjuncts are true of the instance.

To assign identifiers to a new abstract a set features are extracted for each matching phrase and

[1] Specifically, we excluded phrases that began with verbs prepositions, adverbs or determiners; we found this constraint did not affect recall while reducing the number of candidate mentions by more than 50%.

gene id pair just as in training (this constitutes an *instance*) and presented to the classifier for classification. As the classifier returns a *probability* for each instance, the gene id associated with the instance with highest probability is returned as a gene id associated with the abstract, except in the case where the probability is less than some threshold $T, 0 \leq T \leq 1$ in which case no gene id is returned for that phrase.

Training the model involves finding the parameters that maximize the log-likelihood of the training data. As is standard with maximum entropy models we employ a Gaussian prior over the parameters which bias them towards zero to reduce overfitting.

Our model thus has just two parameters which need to be tuned to different datasets (i.e. different organisms): the Gaussian prior and the threshold, T. Tuning the parameters can be done on a held out set (we used the Task 1B development data) or by cross validation:

4 Weakly Supervised Methods for Re-labeling Noisy Normalization Data

The primary contribution of this work is a novel method for re-labeling the noisy training instances within the Task 1B training data sets. Recall that the Task 1B training data were constructed by matching phrases in the abstract against the synonym lists for the gene ids curated for the full text article for which the abstract was written. In many cases, mentions of the gene in the abstract do not appear exactly as they do in the synonym list, which would result in a missed association of that gene id with the abstract. In other cases, the database curators simply did not curate a gene id mentioned in the abstract as it was not relevant to their particular line of interest.

Our method for re-labeling potentially mislabeled instances draws upon existing methods for *weakly supervised learning*. We describe here the generic algorithm and include specific variations below in the experimental setup.

The first step is to partition the training data into two disjoint sets, D_1 and D_2.[2] We then create two instances of the weakly supervised learning problem where in one instance, D_1 is viewed as the labeled training data and D_2 is viewed as the unlabeled data, and in the other instance their roles are reversed. Re-labeling of instances in D_1 is carried out by a classifier or ensemble of classifiers, C_2 trained on D_2. Similarly, instances in D_2 are re-labeled by C_1 trained on D_1. Those instances for which the classifier assigns high confidence (i.e. for which $P(y = "yes" | x)$ is high) but for which the existing label disagrees with the classifier are candidates for re-labeling. Figure 2 diagrams this process below.

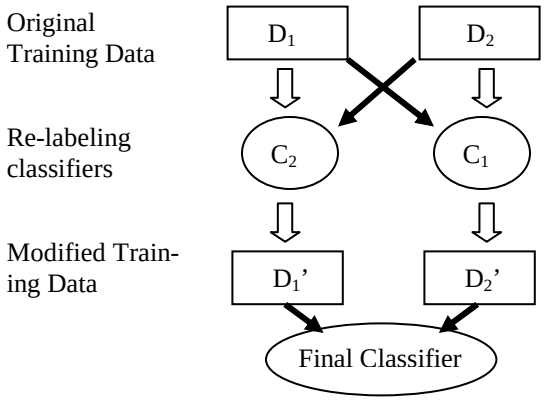


Figure 2. Diagram illustrating the method for re-labeling instances. The solid arrows indicate the training of a classifier from some set of data, while block arrows describe the data flow and re-labeling of instances.

One assumption behind this approach is that not all of the errors in the training data labels are correlated. As such, we would expect that for a particular mislabeled instance in D_1, there may be similar positive instances in D_2 that provide evidence for re-labeling the mislabeled in D_1.

Initial experiments using this approach met with failure or negligible gains in performance. We initially attributed this to too many correlated errors. Detailed error analysis revealed, however, that a significant portion of training instances being re-labeled were derived from matches against the lexicon that were not, in fact, references to genes – i.e. they were other more common English words that happened to appear in the synonym lists for which the classifier mistakenly assigned them high probability.

[2] Note that instances in D_1 and D_2 are also derived form disjoint sets of *abstracts*. This helps ensure that very similar instances are unlikely to appear in different partitions.

Our solution to this problem was to impose a constraint on instances to be re-labeled: The phrase in the abstract associated with the instance *is required to have been tagged as a gene name by a gene name tagger in addition to the instance receiving a high probability by the re-labeling classifier*. Use of a gene name tagger introduces a check against the classifier (trained on the *noisy* training data) and helps to reduce the chance of introducing false positives into the labeled data.

We trained our entity tagger, Carafe, on a the Genia corpus [10] together with the BioCreative Task 1A gene name training corpus. Not all of the entity types annotated in the Genia corpus are genes, however. Therefore we used an appropriate subset of the entity types found in the corpus. Carafe is based on Conditional Random Fields [11] (CRFs) which, for this task, employed a similar set of features to the CRF described in [12].

5 Experiments and Results

The main goal of our experiments was to demonstrate the benefits of re-labeling potentially noisy training instances in the task 1B training data. In this work we focus the weakly supervised re-labeling experiments on the *mouse* data set. In the mouse data there is a strong bias towards false negatives in the training data – i.e. many training instances have a negative label and should have a positive one. Our reasons for focusing on this data are twofold: 1) we believe this situation is likely to be more common in practice since an organism may have impoverished synonym lists or "gaps" in the curated databases and 2) the experiments and resulting analyses are made clearer by focusing on re-labeling instances in one direction only (i.e. from negative to positive).

In this section, we first describe an initial experiment comparing the baseline system (described above) using the original training data with a version trained with an augmented data set where labels changed based on a simple heuristic. We then describe our main body of experiments using various weakly supervised learning methods for re-labeling the data. Finally, we report our overall scores on the evaluation data for all three organisms using the best system configurations derived from the development test data.

5.1 Data and Methodology

We used the BioCreative Task 1B data for all our experiments. For the three data sets, there were 5000 abstracts of training data and 250, 110 and 108 abstracts of development test data for mouse, fly and yeast, respectively. The final evaluation data consisted of 250 abstracts for each organism. In the training data, the ratios of positive to negative *instances* are the following: for mouse: 40279/111967, for fly: 75677/493959 and for yeast: 25108/3856. The number of features in each trained model range from 322110 for mouse, 881398 for fly and 108948 for yeast.

Given a classifier able to rank all the test instances (in our case, the ranks derive from the probabilities output by the maximum entropy classifier), we return only the top n gene identifiers, where n is the number of correct identifiers in the development test data – this results in a balanced F-measure score. We use this metric for all experiments on the development test data as it allows better comparison between systems by factoring out the need to tune the threshold.

On the evaluation data, we do not know n. The system returns a number of identifiers based on the threshold, T. For these experiments, we set T on the development test data and choose three appropriate values for three different evaluation "submissions".

5.2 Experiment Set 1: Effect of match-based re-labeling

Our first set of experiments uses the baseline system described earlier. We compare the results of this system using the Task 1B training data "as provided" with the results obtained by re-labeling some of the negative instances provided to the classifier as positive instances. We re-labeled any instances as positive that matched a gene identifier associated with the abstract regardless of the (potentially incorrect) label associated with the identifier. The Task 1B dataset creators marked an identifier "no" if an exact lexicon match wasn't found in the abstract. As our system matching phase is a bit different (i.e. we remove punctuation and ignore case), this amounts to re-labeling the training data using this looser criterion. The results of this *match-based re-labeling* are shown in Table 1 below.

	Baseline	Re-labeled
Mouse	68.8	72.0
Fly	70.8	75.3
Yeast	89.7	90.0

Table 1 Balanced F-measure scores comparing the baseline vs. a system trained with the match-based re-labeled instances on the development test data.

5.3 Experiment Set 2: Effect of Weakly Supervised Re-labeling

In our next set of experiments we tested a number of different weakly supervised learning configurations. These different methods simply amount to different rankings of the instances to re-label (based on confidence and the gene name tags). The basic algorithm (outlined in Figure 1) remains the same in all cases. Specifically, we investigated three methods for ranking the instances to re-label: 1) naïve self-training, 2) self-training with bagging, and 3) co-training.

Naïve self-training consisted of training a single maximum entropy classifier with the full feature set on each partition and using it to re-label instances from the other partition based on confidence.

Self training with bagging followed the same idea but used bagging. For each partition, we trained 20 separate classifiers on random subsets of the training data using the full feature set. The confidence assigned to a test instance was then defined as the product of the confidences of the individual classifiers.

Co-training involved training two classifiers for each partition with feature split. We split the features into *context-based* features such as the surrounding words and the number of gene ids matching the current phrase, and *lexically-based* features that included the phrase itself, affixes, the number of tokens in the phrase, etc. We computed the aggregated confidences for each instance as the product of the confidences assigned by the resulting context-based and lexically-based classifiers.

We ran experiments for each of these three options both *with* the gene tagger and *without* the gene tagger. The systems that included the gene tagger ranked all instances derived from tagged phrases above all instances derived from phrases that were not tagged regardless of the classifier confidence.

A final experimental condition we explored was comparing batch re-labeling vs. incremental re-labeling. Batch re-labeling involved training the classifiers once and re-labeling all *k* instances using the same classifier. Incremental re-labeling consisted of iteratively re-labeling *n* instances over *k/n* epochs where the classifiers were re-trained on each epoch with the newly re-labeled training data. Interestingly, incremental re-labeling did not perform better than batch re-labeling in our experiments. All results reported here, therefore, used batch re-labeling.

After the training data was re-labeled, a single maximum entropy classifier was trained on the entire (now re-labeled) training set. This resulting classifier was then applied to the development set in the manner described in Section 3.

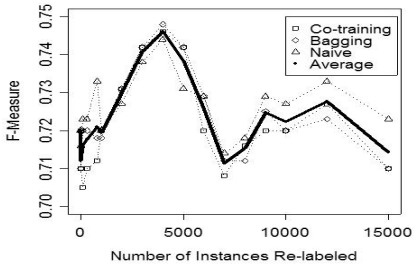
MAX	With Tagger	Without Tagger
Self-Naïve	74.4 (4000)	72.3 (5000)
Self-Bagging	74.8 (4000)	73.5 (6000)
Co-Training	74.6 (4000)	72.7 (6000)

AVG	With Tagger	Without Tagger
Self-Naïve	72.2	71.2
Self-Bagging	72.2	71.5
Co-Training	71.9	71.2

Table 2. Maximum and average balanced f-measure scores on the mouse data set for each of the six system configurations for all values of *k* – the number of instances re-labeled. The numbers in parentheses indicate for which value of *k* the maximum value was achieved.

We tested each of these six configurations for different values of *k*, where *k* is the total number of instances re-labeled[3]. Table 2 highlights the maximum and average balanced f-measure scores across all values of *k* for the different system configurations. Both the maximum and averaged scores appear noticeably higher when constraining the instances to re-label with the tagger. The three weakly supervised methods perform comparably with bagging performing slightly better.

[3] The values of *k* considered here were: 0, 10, 20, 50, 100, 200, 300, 500, 800, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 12000 and 15000.



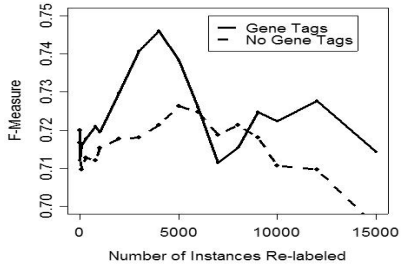


Figure 3. The top graph shows balanced F-measure scores against the number of instances re-labeled when using the tagger as a constraint. The bottom graph compares the re-labeling of instances with the gene tagger as a constraint and without.

In order to gain further insight into re-labeling instances, we have plotted the balanced F-measure performance on the development test for various values of *k*. The upper graph indicates that the three different methods correlate strongly. The bottom graph makes apparent the benefits of tagging as a constraint. It also points to the weakness of the tagger, however. At *k*=7000 and *k*=8000, the system tends to perform *worse* when using the tags as a constraint. This indicates that tagger recall errors have the potential to filter out good candidates for re-labeling.

Another observation from the graphs is that performance actually drops for small values of *k*. This would imply that many of the instances the classifiers are most confident about re-labeling are in fact spurious. To support this hypothesis, we trained the baseline system on the entire training set and computed its *calibration error* on the development test data. The calibration error measures how "realistic" the probabilities output by the classifier are. See [13] for details.

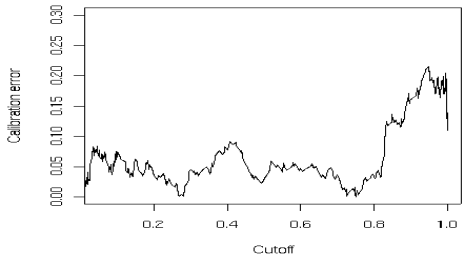


Figure 4. Classifier calibration error on the development test data.

Figure 4 illustrates the estimated calibration error at different thresholds. As can be seen, the error is greatest for high confidence values indicating that the classifier is indeed very confidently predicting an instance as positive when it is negative. Extrapolating this calibration error to the re-labling classifiers (each trained on one half of the training data) offers some explanation as to why re-labeling starts off so poorly. The error mass is exactly where we do not want it - at the highest confidence values. This also offers an explanation as to why incremental re-labeling did not help. Fortunately, introducing a gene tagger as a constraint mitigates this problem.

5.4 Experiment Set 3: Final Evaluation

We report our results using the best overall system configurations on the Task 1B evaluation data. We "submitted" 3 runs for two different *mouse* configurations and one for both *fly* and *yeast*. The highest scores over the 3 runs are reported in Table 3. *MouseWS* used the best weakly supervised method as determined on the development test data: *bagging* with k=4000. *MouseMBR*, *YeastMBR* and *FlyMBR* used match-based re-labeling described in Section 5.2. The Gaussian prior was set to 2.0 for all runs and the 3 submissions for each configuration only varied in the threshold value *T*.

	F-measure	Precision	Recall
MouseWS	0.784	0.81	0.759
MouseMBR	0.768	0.795	0.743
FlyMBR	0.767	0.767	0.767
YeastMBR	0.902	0.945	0.902

Table 3. Final evaluation results.

These results are competitive compared with the BioCreAtIvE Task 1B results where the highest F-measures for mouse, fly and yeast were 79.1, 81.5 and 92.1 with the medians at 73.8, 66.1 and 85.8, respectively. The results for mouse and fly improve upon previous best reported results with an organism invariant, automatic system [1].

6 Conclusions

The quality of training data is paramount to the success of fully automatic, organism invariant approaches to the normalization problem. In this paper we have demonstrated the utility of weakly supervised learning methods in conjunction with a gene name tagger for re-labeling noisy training data for gene name normalization. The result being higher quality data with corresponding higher performance on the BioCreAtIvE Task 1B gene name normalization task.

Future work includes applying method outlined here for correcting noisy data to other classification problems. Doing so generally requires an independent "filter" to restrict re-labeling – the equivalent of the gene tagger used here. We also have plans to improve classifier calibration. Integrating confidence estimates produced by the gene name tagger, following [14], is another avenue for investigation.

Acknowledgements

We thank Alex Morgan, Lynette Hirschman, Marc Colosimo, Jose Castano and James Pustejovsky for helpful comments and encouragement. This work was supported under MITRE Sponsored Research 51MSR123-A5.

References

1. Crim, J., R. McDonald, and F. Pereira. *Automatically Annotating Documents with Normalized Gene Lists*. in *EMBO Workshop - A critical assessment of text mining methods in molecular biology*. 2004. Granada, Spain.
2. Blum, A. and T. Mitchell. *Combining Labeled and Unlabeled Data with Co-training*. 1998. Proceedings of the Workshop on Computational Learning Theory: Morgan Kaufmann.
3. Banko, M. and E. Brill. *Scaling to very very large corpora for natural language disambiguation*. in *ACL/EACL*. 2001.
4. Ng, V. and C. Cardie. *Weakly Supervised Natural Language Learning Without Redundant Views*. in *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*. 2003.
5. Hirschman, L., et al., *Overview of BioCreAtIvE task 1B: Normalized Gene Lists.* BioMed Central BioInformatics, 2005(Special Issue on BioCreAtIvE).
6. Craven, M. and J. Kumlien, *Constructing Biological Knowledge Bases by Extracting Information from Text Sources.* 1999: p. 77-86.
7. Morgan, A., et al., *Gene Name Extraction Using FlyBase Resources.* ACL Workshop on Natural Language Processing in Biomedicine, 2003.
8. Morgan, A.A., et al., *Gene name identification and normalization using a model organism database.* J Biomed Inform, 2004. **37**(6): p. 396-410.
9. Nigam, K. and R. Ghani. *Analyzing the effectiveness and applicability of co-training*. in *Information and Knowledge Management*. 2000.
10. Kim, J.-D., et al., *GENIA Corpus -- a semantically annotated corpus for bio-text mining.* Bioinformatics, 2003. **19**((Supppl 1)): p. 180-182.
11. Lafferty, J., A. McCallum, and F. Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. in *18th International Conf. on Machine Learning*. 2001. San Francisco, CA: Morgan Kaufmann.
12. McDonald, R. and F. Pereira. *Identifying Gene and Protein Mentions in Text Using Conditional Random Fields*. in *A critical assessment of text mining methods in molecular biology, BioCreative 2004*. 2004. Grenada, Spain.
13. Cohen, I. and M. Goldszmidt. *Properties and Benefits of Calibrated Classifiers*. in *EMCL/PKDD*. 2004. Pisa, Italy.
14. Culotta, A. and A. McCallum. *Confidence Estimation for Information Extraction*. in *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. 2004. Boston, MA.