Active Annotation

Andreas Vlachos
William Gates Building
Computer Laboratory
University of Cambridge
av308@cl.cam.ac.uk

Abstract

This paper introduces a semi-supervised learning framework for creating training material, namely active annotation. The main intuition is that an unsupervised method is used to initially annotate imperfectly the data and then the errors made are detected automatically and corrected by a human annotator. We applied active annotation to named entity recognition in the biomedical domain and encouraging results were obtained. The main advantages over the popular active learning framework are that no seed annotated data is needed and that the reusability of the data is maintained. In addition to the framework, an efficient uncertainty estimation for Hidden Markov Models is presented.

1 Introduction

Training material is always an issue when applying machine learning to deal with information extraction tasks. It is generally accepted that increasing the amount of training data used improves performance. However, training material comes at a cost, since it requires annotation.

As a consequence, when adapting existing methods and techniques to a new domain, researchers and users are faced with the problem of absence of annotated material that could be used for training. A good example is the biomedical domain, which has attracted the attention of the NLP community relatively recently (Kim et al., 2004). Even though there are plenty of biomedical texts, very little of it is annotated, such as the GENIA corpus (Kim et al., 2003).

A very popular and well investigated framework in order to cope with the lack of training material is the active learning framework (Cohn et al., 1995; Seung et al., 1992). It has been applied to various NLP/IE tasks, including named entity recognition (Shen et al., 2004) and parse selection (Baldridge and Osborne, 2004) with rather impressive results in reducing the amount of annotated training data. However, some criticism of active learning has been expressed recently, concerning the reusability of the data (Baldridge and Osborne, 2004).

This paper presents a framework in order to deal with the lack of training data for NLP tasks. The intuition behind it is that annotated training data is produced by applying an (imperfect) unsupervised method, and then the errors inserted in the annotation are detected automatically and reannotated by a human annotator. The main difference compared to active learning is that instead of selecting unlabeled instances for annotation, possible erroneous instances are selected for checking and correction if they are indeed erroneous. We will refer to this framework as "active annotation" in the rest of the paper.

The structure of this paper is as follows. In Section 2 we describe the software and the dataset used. Section 3 explores the effect of errors in the training data and motivates the active annotation framework.

In Section 4 we describe the framework in detail, while Section 5 presents a method for estimating uncertainty for HMMs. Section 6 presents results from applying the active annotation. Section 7 compares the proposed framework to active learning and Section 8 attempts an analysis of its performance. Finally, Section 9 suggests some future work.

2 Experimental setup

The data used in the experiments that follow are taken from the BioNLP 2004 named entity recognition shared task (Kim et al., 2004). The text passages have been annotated with five classes of entities, "DNA", "RNA", "protein", "cell_type" and "cell_line". In our experiments, following the example of Dingare et al. (2004), we simplified the annotation to one entity class, namely "gene", which includes the DNA, RNA and protein classes. In order to evaluate the performance on the task, we used the evaluation script supplied with the data, which computes the F-score ($F_1 = \frac{2*Precision*Recall}{Precision+Recall}$) for each entity class. It must be noted that all tokens of an entity must be recognized correctly in order to count as a correct prediction. A partially recognized entity counts both as a precision and recall error. In all the experiments that follow, the official split of the data in training and testing was maintained.

The named entity recognition system used in our experiments is the open source NLP toolkit Lingpipe[1]. The named entity recognition module is an HMM model using Witten-Bell smoothing. In our experiments, using the data mentioned earlier it achieved 70.06% F-score.

3 Effect of errors

Noise in the training data is a common issue in training machine learning for NLP tasks. It can have significant effect on the performance, as it was pointed out by Dingare et al. (2004), where the performance of the same system on the same task (named entity recognition in the biomedical domain) was lower when using noisier material. The effect of noise in the data used to train machine learning algorithms for NLP tasks has been explored by Osborne (2002), using the task of shallow parsing as the case study and a variety of learners. The impact of different types of noise was explored and learner specific extensions were proposed in order to deal with it.

In our experiments we explored the effect of noise in training the selected named entity recognition system, keeping in mind that we are going to use an unsupervised method to create the training material. The kind of noise we expect is mislabelled instances. In order to simulate the behavior of a hypothetical unsupervised method, we corrupted the training data artificially using the following models:

- LowRecall: Change tokens labelled as entities to non-entities. It must be noted that in this model, due to the presence of multi-token entities precision is reduced too, albeit less than recall.
- LowRecall_WholeEntities: Change the labeling of whole entities to non-entities. In this model, precision is kept intact.
- LowPrecision: Change tokens labelled as non-entities to entities.
- Random: Entities and non-entities are changed randomly. It can be viewed alternatively as a random tagger which labels the data with some accuracy.

The level of noise inserted is adjusted by specifying the probability with which a candidate label is changed. In all the experiments in this paper, for a particular model and level of noise, the corruption of the dataset was repeated five times in order to produce more reliable results. In practice, the behavior of an unsupervised method is likely to be a mixture of the above models. However, given that the method would tag the data with a certain performance, we attempted through our experiments to identify which of these (extreme) behaviors would be less harmful. In Figure 1, we present graphs showing the effect of noise inserted with the above models. The experimental procedure was to add noise to the training data according to a model, evaluate the performance of the hypothetical tagger that produced it, train Lingpipe on the noisy training data and evaluate the performance of the latter on the test data. The process was repeated for various levels of noise. In the top graph, the F-score achieved by Lingpipe (F-ling) is plotted against the F-score of

[1] http://www.alias-i.com/lingpipe/

the hypothetical tagger (F-tag), while in the bottom graph the F-score achieved by Lingpipe (F-ling) is plotted against the number of erroneous classifications made by the hypothetical tagger.

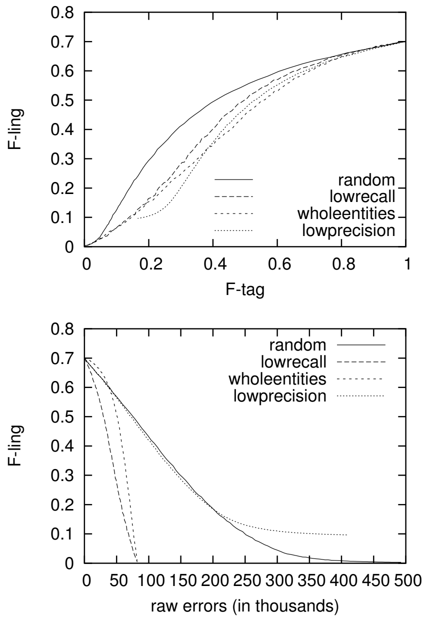

Figure 1: F-score achieved by Lingpipe is plotted against (a) the F-score of the hypothetical tagger in the top graph and (b) the number of errors made by the hypothetical tagger in the bottom graph.

A first observation is that limited noise does not affect the performance significantly, a phenomenon that can be attributed to the capacity of the machine learning method to deal with noise. From the point of view of correcting mistakes in the training data this suggests that not all mistakes need to be corrected. Another observation is that while the performance for all the models follow similar curves when plotted against the F-score of the hypothetical tagger, the same does not hold when plotted against the number of errors. While this can be attributed to the unbalanced nature of the task (very few entity tokens compared to non-entities), it also suggests that the raw number of errors in the training data is not a good indicator for the performance obtained by training on it. However, it represents the effort required to obtain the maximum performance from the data by correcting it.

4 Active Annotation

In this section we present a detailed description of the active annotation framework. Initially, we have a pool of unlabeled data, D, whose instances are annotated using an unsupervised method u, which does not need training data. As expected, a significant amount of errors is inserted during this process. A list L is created containing the tokens that have not been checked by a human annotator. Then, a supervised learner s is used to train a model M on this noisy training material. A query module q, which uses the model created by s decides which instances of D will be selected to be checked for errors by a human annotator. The selected instances are removed from L so that q does not select them again in future. The learner s is then trained on this partially corrected training data and the sequence is repeated from the point of applying the querying module q. The algorithm written in pseudocode appears in Figure 2.

```
Data $D$, unsupervised tagger $u$,
supervised learner $s$, query module $q$.
Initialization:
    Apply $u$ to $D$.
    Create list of instances $L$.
Loop:
    Using $s$ train a model $M$ on $D$.
    Using $q$ and $M$ select a batch of instances $B$
    to be checked.
    Correct the instances of $B$ in $D$.
    Remove the instances of $B$ from $L$.
Repeat until:
    $L$ is empty or annotator stops.
```

Figure 2: Active annotation algorithm

Comparing it with active learning, the similarities are apparent. Both frameworks have a loop in which a query module q, using a model produced by the learner, selects instances to be presented to a human annotator. The efficiency of active annotation can be measured in two ways, both of them used in evaluating active learning. The first is to measure the reduction in the checked instances needed in order to achieve a certain level of performance. The second is the increase in performance for a fixed number of checked instances. Following the active learning paradigm, a baseline for active annotation is random selection of instances to be checked.

There are though some notable differences. During initialization, an unsupervised method u is required to provide an initial tagging on the data D. This is an important restriction which is imposed by the lack of any annotated data. Even under this restriction, there are some options available, especially for tasks which have compiled resources. One option is to use an unsupervised learning algorithm, such the one presented by Collins & Singer (1999), where a seed set of rules is used to bootstrap a rule-based named entity recognizer. A different approach could be the use of a dictionary-based tagger, as in Morgan et al. (2003). It must be noted that the unsupervised method used to provide the initial tagging does not need to generalize to any data (a common problem for such methods), it only needs to perform well on the data used during active annotation. Generalization on unseen data is an attribute we hope that the supervised learning method s will have after training on the annotated material created with active annotation.

The query module q is also different from the corresponding module in active learning. Instead of selecting unlabeled informative instances to be annotated and added to the training data, its purpose is to identify likely errors in the imperfectly labelled training data, so that they are checked and corrected by the human annotator.

In order to perform error-detection, we chose to adapt the approach of Nakagawa and Matsumoto (2002) which resembles uncertainty based sampling for active learning. According to their paradigm, likely errors in the training data are instances that are “hard” for the classifier and inconsistent with the rest of the data. In our case, we used the uncertainty of the classifier as the measure of the “hardness” of an instance. As an indication of inconsistency, we used the disagreement of the label assigned by the classifier with the current label of the instance. Intuitively, if the classifier disagrees with the label of an instance used in its training, it indicates that there have been other similar instances in the training data that were labelled differently. Returning to the description of active annotation, the query module q ranks the instances in L first by their inconsistency and then by decreasing uncertainty of the classifier. As a result, instances that are inconsistent with the rest of the data and hard for the classifier are selected first, then those that are inconsistent but easy for the classifier, then the consistent ones but hard for the classifier and finally the consistent and easy ones.

While this method of detecting errors resembles uncertainty sampling, there are other approaches that could have been used instead and they can be very different. Sjöbergh and Knutsson (2005) inserted artificial errors and trained a classifier to recognize them. Dickinson and Meuers (2003) proposed methods based on n-grams occurring with different labellings in the corpus. Therefore, while it is reasonable to expect some correlation between the selections of active annotation and active learning (hard instances are likely to be erroneously annotated by the unsupervised tagger), the task of selecting hard instances is quite different from detecting errors. The use of the disagreement between taggers for selecting candidates for manual correction is reminiscent of corrected co-training (Pierce and Cardie, 2001). However, the main difference is corrected co-training results in a manually annotated corpus, while active annotation allows automatically annotated instances to be kept.

5 HMM uncertainty estimation

In order to perform error detection according to the previous section we need to obtain uncertainty estimations over each token from the named entity recognition module of Lingpipe. For each token t and possible label l, Lingpipe estimates the following Hidden Markov Model from the training data:

$$P(t[n], l[n] | l[n-1], t[n-1], t[n-2]) \qquad (1)$$

When annotating a certain text passage, the tokens are fixed and the joint probability of Equation 1 is computed for each possible combination of labels. From Bayes’ rule, we obtain:

$$P(l[n]|t[n], l[n-1], t[n-1], t[n-2]) = \\ \frac{P(t[n], l[n]|l[n-1], t[n-1], t[n-2])}{P(t[n]|l[n-1], t[n-1], t[n-2])} \qquad (2)$$

For fixed token sequence $t[n], t[n-1], t[n-2]$ and previous label ($l[n-1]$) the second term of the left part of Equation 2 is a fixed value. Therefore, under these conditions, we can write:

$$P(l[n]|t[n], l[n-1], t[n-1], t[n-2]) \propto \\ P(t[n], l[n]|l[n-1], t[n-1], t[n-2]) \quad (3)$$

From Equation 3 we obtain an approximation for the conditional distribution of the current label ($l[n]$) conditioned on the previous label ($l[n-1]$) for a fixed sequence of tokens. It must be stressed that the later restriction is very important. The resulting distribution from Equation 3 cannot be compared across different token sequences. However, for the purpose of computing the uncertainty over a fixed token sequence it is a reasonable approximation. One way to estimate the uncertainty of the classifier is to calculate the conditional entropy of this distribution. The conditional entropy for a distribution $P(X|Y)$ can be computed as:

$$H[X|Y] = \sum_y P(Y = y) \sum_x logP(X = x|Y = y) \quad (4)$$

In our case, X is $l[n]$ and Y is $l[n-1]$. Function 4 can be interpreted as the weighted sum of the entropies of $P(l[n]|l[n-1])$ for each value of $l[n-1]$, in our case the weighted sum of entropies of the distribution of the current label for each possible previous label. The probabilities for each tag (needed for $P(l[n-1])$) are not calculated directly from the model. $P(l[n])$ corresponds to $P(l[n]|t[n], t[n-1], t[n-2])$, but since we are considering a fixed token sequence, we approximate its distribution using the conditional probability $P(l[n]|t[n], l[n-1], t[n-1], t[n-2])$, by marginalizing over $l[n-1]$.

Again, it must be noted that the above calculations are to be used in estimating uncertainty over a single word. One property of the conditional entropy is that it estimates the uncertainty of the predictions for the current label given knowledge of the previous tag, which is important in our application because we need the uncertainty over each label independently from the rest of the sequence. This is confirmed by the theory, from which we know that for a conditional distribution of X given Y the following equation holds, $H[X|Y] = H[X, Y] - H[Y]$, where H denotes the entropy.

A different way of obtaining uncertainty estimations from HMMs in the framework of active learning has been presented in (Scheffer et al., 2001). There, the uncertainty is estimated by the margin between the two most likely predictions that would result in a different current label, explicitly:

$$M = max_{i,j} \{P(t[n] = i|t[n-1] = j)\} - \\ max_{k,l,k \neq i} \{P(t[n] = k|t[n-1] = l)\} \quad (5)$$

Intuitively, the margin M is the difference between the two highest scored predictions that disagree. The lower the margin, the higher the uncertainty of the HMM on the token at question. A drawback of this method is that it doesn't take into account the distribution of the previous label. It is possible that the two highest scored predictions are obtained for two different previous labels. It may also be the case that a highly scored label can be obtained given a very improbable previous label. Finally, an alternative that we did not explore in this work is the Field Confidence Estimation (Culotta and McCallum, 2004), which allows the estimation of confidence over sequences of tokens, instead of singleton tokens only. However, in this work confidence estimation over singleton tokens is sufficient.

6 Experimental Results

In this section we present results from applying active annotation to biomedical named entity recognition. Using the noise models described in Section 3, we corrupted the training data and then using Lingpipe as the supervised learner we applied the algorithm of Figure 2. The batch of tokens selected to be checked in each round was 2000 tokens. As a baseline for comparison we used random selection of tokens to be checked. The results for various noise models and levels are presented in the graphs of Figure 3. In each of these graphs, the performance of Lingpipe trained on the partially corrected material (F-ling) is plotted against the number of checked instances, under the label “entropy”.

In all the experiments, active annotation significantly outperformed random selection, with the exception of 50% Random, where the high level of noise (the F-score of the hypothetical tagger that provided the initial data was 0.1) affected the initial

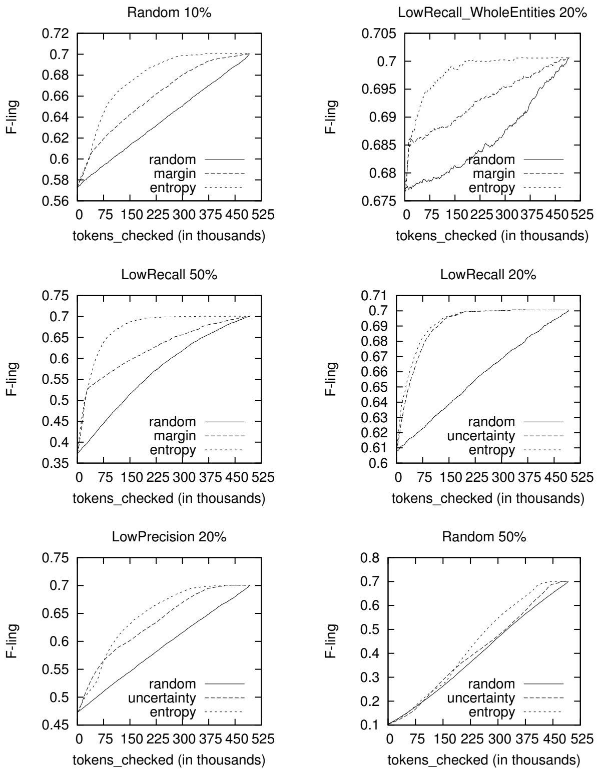

Figure 3: F-score achieved by Lingpipe is plotted against the number of checked instances for various models and levels of noise.

judgements of the query module on which instances should be checked. After having checked some portion of the dataset though, active annotation started outperforming random selection. In the graphs for the 10% Random, 20% LowRecall_WholeEntities and 50% LowRecall noise models, under the label "margin", appear the performance curves obtained using the uncertainty estimation of Scheffer et al. (2001). Even though active annotation using this method performs better than random selection, active annotation using conditional entropy performs significantly better. These results provide evidence of the theoretical advantages of conditional entropy described earlier. We also ran experiments using pure uncertainty based sampling (i.e. without checking the consistency of the labels) on selecting instances to be checked. The performance curves appear under the label "uncertainty" for the 20% LowRecall, 50% Random and 20% LowPrecision noise models. The uncertainty was estimated using the method described in Section 5. As expected, uncertainty based sampling performed reasonably well, better than random selection but worse than using labelling consistency, except for the initial stage of 20% LowPrecision.

7 Active Annotation versus Active Learning

In order to compare active annotation to active learning, we run active learning experiments using the same dataset and software. The paradigm employed was uncertainty based sampling, using the uncertainty estimation presented in Sec. 5. HMMs require annotated sequences of tokens, therefore annotating whole sentences seemed as the natural choice, as in (Becker et al., 2005). While token-level selections could be used in combination with EM, (as in (Scheffer et al., 2001)), constructing a corpus of individual tokens would result in a corpus that would be very difficult to be reused, since it would be partially labelled. We employed the two standard options of selecting sentences, selecting the sentences with the highest average uncertainty over the tokens or selecting the sentence containing the most uncertain token. As cost metric we used the number of tokens, which allows more straightforward comparison with active annotation.

In Figure 4 (left graph), each active learning experiment is started by selecting a random sentence as seed data, repeating the seed selection 5 times. The random selection is repeated 5 times for each seed selection. As in (Becker et al., 2005), selecting the sentences with the highest average uncertainty (ave) performs better than selecting those with the most uncertain token (max).

In the right graph, we compare the best active learning method with active annotation. Apparently, the performance of active annotation is highly dependent on the performance of the unsupervised tagger used to provide us with the initial annotation of the data. In the graph, we include curves for two of the noise models reported in the previous section, LowRecall20% and LowRecall50% which correspond to tagging performance of 0.66 / 0.69 / 0.67 and 0.33 / 0.43 / 0.37 respectively, in terms of Recall / Precision / F. We consider such tagging performances feasible with a dictionary-based tagger, since Morgan et al. (2003) report performance of

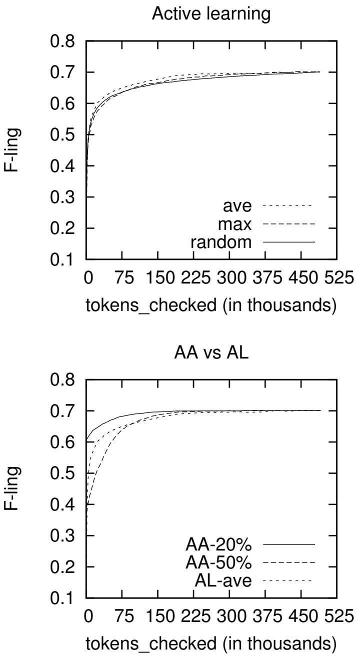

Figure 4: Left, comparison among various active learning methods. Right, comparison of active learning and active annotation.

0.88 / 0/78 / 83 with such a method.

These results demonstrate that active annotation, given a reasonable starting point, can achieve reductions in the annotation cost comparable to those of active learning. Furthermore, active annotation produces an actual corpus, albeit noisy. Active learning, as pointed out by Baldridge & Osborne (2004), while it reduces the amount of training material needed, it selects data that might not be useful to train a different learner. In the active annotation framework, it is likely to preserve correct instances that might not be useful to the machine learning method used to create it, but maybe beneficial to a different method. Furthermore, producing an actual corpus can be very important when adding new features to the model. In the case of biomedical NER, one could consider adding document-level features, such as whether a token has been seen as part of a gene name earlier in the document. With the corpus constructed using active learning this is not feasible, since it is unlikely that all the sentences of a document are selected for annotation. Also, if one intended to use the same corpus for a different task, such as anaphora resolution, again the imperfectly annotated corpus constructed using active annotation can be used more efficiently than the partially annotated one produced by active learning.

8 Selecting errors

In order to investigate further the behavior of active annotation, we evaluated the performance of the trained supervised method against the number of errors corrected by the human annotator. The aim of this experiment was to verify whether the improvement in performance compared to random selection is due to selecting "informative" errors to correct, or due to the efficiency of the error detection technique.

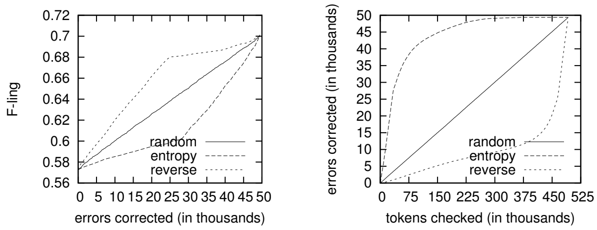

Figure 5: Left: F-score achieved by Lingpipe is plotted against the number of corrected errors. Right: Errors corrected plotted against the number of checked tokens.

In Figure 5, we present such graphs for the 10% Random noise model. Similar results were obtained with different noise models. As can be observed on the left graph, the errors corrected initially during random selection are far more informative compared to those corrected at the early stages of active annotation (labelled "entropy"). The explanation for this is that using the error detection method described in Section 4, the errors that are detected are those on which the supervised method s disagrees with the training material, which implies that even if such an instance is indeed an error then it didn't affect s. Therefore, correcting such errors will not improve the performance significantly. Informative errors are those that s has learnt to reproduce with high certainty. However, such errors are hard to detect because similar attributes are exhibited usually by correctly labelled instances. This can be verified by the curves labelled "reverse" in the graphs of Figure 5, in which the ranking of the instances to be selected was reversed, so that instances where the supervised method agrees confidently with the training material are selected first. The fact that errors with high uncertainty are less informative than those with low uncertainty suggests that active annotation, while being related to active learning, it is sufficiently different. The right graph suggests that the error-detection performance during active annotation is much better than that of random selection. Therefore, the performance of active annotation could be improved by preserving the high error-detection performance and selecting more informative errors.

9 Future work

This paper described active annotation, a semi-supervised learning framework that reduces the effort needed to create training material, which is very important in adapting existing trainable methods to new domains. Future work should investigate the applicability of the framework in a variety of NLP/IE tasks and settings. We intend to apply this framework to NER for biomedical literature from the FlyBase project for which no annotated datasets exist.

While we have used the number of instances checked by a human annotator to measure the cost of annotation, this might not be representative of the actual cost. The task of checking and possibly correcting instances differs from annotating them from scratch. In this direction, experiments in realistic conditions with human annotators should be carried out. We also intend to explore the possibility of grouping similar mistakes detected in a round of active annotation, so that the human annotator can correct them with less effort. Finally, alternative error-detection methods should be investigated.

Acknowledgments

The author was funded by BBSRC, grant number 38688. I would like to thank Ted Briscoe and Bob Carpenter for their feedback and comments.

References

J. Baldridge and M. Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

M. Becker, B. Hachey, B. Alex, and C. Grover. 2005. Optimising selective sampling for bootstrapping named entity recognition. In *Proceedings of the Workshop on Learning with Multiple Views, ICML.*

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. 1995. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, volume 7.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on EMNLP and VLC.*

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proceedings of HLT 2004*, Boston, MA.

M. Dickinson and W. D. Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of EACL 2003*, pages 107–114, Budapest, Hungary.

S. Dingare, J. Finkel, M. Nissim, C. Manning, and C. Grover. 2004. A system for identifying named entities in biomedical text: How results from two evaluations reflect on both the system and the evaluations. In *The 2004 BioLink meeting at ISMB.*

J. D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. Genia corpus - a semantically annotated corpus for biotextmining. In *ISMB (Supplement of Bioinformatics).*

J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier, editors. 2004. *Proceedings of JNLPBA, Geneva.*

A. Morgan, L. Hirschman, A. Yeh, and M. Colosimo. 2003. Gene name extraction using FlyBase resources. In *Proceedings of the ACL 2003 Workshop on NLP in Biomedicine*, pages 1–8.

T. Nakagawa and Y. Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of COLING 2002.*

M. Osborne. 2002. Shallow parsing using noisy and non-stationary training material. *J. Mach. Learn. Res.*, 2:695–719.

D. Pierce and C. Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *Proceedings of EMNLP 2001*, pages 1–9.

T. Scheffer, C. Decomain, and S. Wrobel. 2001. Active hidden Markov models for information extraction. *Lecture Notes in Computer Science*, 2189:309+.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of COLT 1992.*

D. Shen, J. Zhang, J. Su, G. Zhou, and C. L. Tan. 2004. Multi-criteria-based active learning for named entity recongition. In *Proceedings of ACL 2004*, Barcelona.

J. Sjöbergh and O. Knutsson. 2005. Faking errors to avoid making errors: Machine learning for error detection in writing. In *Proceedings of RANLP 2005.*