Using Prediction from Sentential Scope to Build a Pseudo Co-Testing Learner for Event Extraction

Shasha Liao
Computer Science Department
New York University
liaoss@cs.nyu.edu

Ralph Grishman
Computer Science Department
New York University
grishman@cs.nyu.edu

Abstract

Event extraction involves the identification of instances of a type of event, along with their attributes and participants. Developing a training corpus by annotating events in text is very labor intensive, and so selecting informative instances to annotate can save a great deal of manual work. We present an active learning (AL) strategy, *pseudo co-testing*, based on one view from a classifier aiming to solve the original problem of event extraction, and another view from a classifier aiming to solve a coarser granularity task. As the second classifier can provide more graded matching from a wider scope, we can build a set of pseudo-contention-points which are very informative, and can speed up the AL process. Moreover, we incorporate multiple selection criteria into the pseudo co-testing, seeking training examples that are informative, representative, and varied. Experiments show that pseudo co-testing can reduce annotation labor by 81%; incorporating multiple selection criteria reduces the labor by a further 7%.

1 Introduction

The goal of event extraction is to identify instances of a class of events in text. There can be many event types; for example, the ACE 2005 event extraction task involved a set of 33 generic event types and subtypes appearing frequently in the news. A typical event extraction task, in addition to identifying the event itself, also identifies all of the participants and attributes of the event; these are the entities that are involved in that event. Annotating a corpus in order to train an event tagger is a costly task.

First of all, event extraction is difficult and requires substantial training data. The same event might be presented in various expressions, and an expression might represent different events in different contexts. For example, "retire" and "resign" can both represent an *End-Position* event, while "leave" can represent either an *End-Position* or *Move* event in different contexts. Moreover, for each event type, the event participants and attributes may also appear in multiple forms and exemplars of the different forms may be required.

Furthermore, compared to other tasks like name tagging or part of speech tagging, events of a particular type appear relatively rarely in a document. One document might only contain one or two events of a given type, or even none at all. For the ACE 2005 event extraction task, *Attack* events have the highest frequency in the training corpus (2240 times, an average of 4 events per document), while *Start-Position* events only appear 232 times (an average of 1/3 event per document). As a result, to acquire enough training samples, we need to annotate a lot of documents. If we can predict which documents, or even which sentences to annotate, we can save a lot of time.

Considering the complexity of event extraction and the labor of annotating an event, providing the annotator with an informative sample to annotate is especially important. Active learning (AL) is a good way to do so because it aims to keep the human annotation effort to a minimum, only asking for advice where the training utility of the result of such a query is high.

Active learning is a supervised machine learning technique in which the learner is in control of the selection of data used for learning. The intent is to ask an *oracle* - typically a human with extensive knowledge of the domain at hand

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 714–722,
Chiang Mai, Thailand, November 8 – 13, 2011.

- about the classes of instances for which the model trained so far makes unreliable predictions. Selective sampling methods, introduced by Cohn, Atlas and Ladner (1994), made the learner query the oracle about data that is likely to be misclassified. This is the crucial aspect of AL – finding "good" instances for a human to annotate.

In this paper, we investigate the use of AL in event extraction. In particular, we apply the active learning approach to the *Attack* event, because it is the most frequent event type in the ACE corpus, and is particularly challenging because of the large number of different expressions: there are 312 different words in the corpus that serve at least once as the main word (the "trigger") of an *Attack* event,

After studying several sampling strategies, we settled upon a *pseudo co-testing* approach where a second classifier which solves a coarser variant of the original task is used. Furthermore, we incorporate multiple selection criteria into the pseudo co-testing, not only selecting more informative sentences, but also considering their distribution in the sample pool, and the diversity of the instances added to the training set at the same time.

2 Event Extraction

2.1 Task Description

ACE defines an event as a specific occurrence involving participants[1], and it annotates 8 types and 33 subtypes of events. In this task, an *event mention* is a phrase or sentence within which an event is described, including trigger and arguments. An event mention must have one and only one trigger, and can have an arbitrary number of arguments. The *event trigger* is the main word that most clearly expresses an event occurrence. The *event mention arguments (roles)*[2] are the *Entity/Timex mentions*[3] that are involved in an event mention, and their relation to the event. For example, an *Attack* event might include participants like *Attacker* or *Target*, or attributes like *Time-Within* and *Place*. Arguments will be taggable only when they occur within the scope of the corresponding event, typically the same sentence.

Consider the sentence:

(1) This Friday in France, ***Bob Cole*** *was on his way home when* ***he*** *was* ***attacked****…*

Event extraction depends on previous phases like name identification, entity mention classification, and entity mention coreference. Table 1 shows the results of this preprocessing. Note that entity mentions that share the same EntityID are coreferential and treated as the same object.

Entity/Timex mention		**Head**	**Entity ID**	**Type**
1-1-1		France	1-1	GPE
1-T1-1		Friday	1-T1	Timex
1-2-1		Bob Cole	1-2	PER
1-2-2		He	1-2	PER
Event type	**Trigger**	**Role**		
		Place	**Target**	**Time**
Attack	attacked	1-1-1	1-2-2	1-T1-1

Table 1. An example of Entity /Timex mentions, and Attack events

In this example, there is one *Attack* event, which contains attributes and participants including place, target and time.

2.2 Baseline Event Tagger

Identifying a potential trigger – the word most clearly expressing the event – is essential for event extraction. Usually, the trigger itself is the most important clue in detecting and classifying the type of an event; for example, words like "attack", "conflict", and "beat" are more likely to represent an *Attack* event, while "meet", "eat", and "shopping" are not likely to be triggers of *Attack* events. Then, once we find possible triggers, we can apply the argument / role identification to find the participants or attributes of the event. For example, the subject of the trigger word "attack" is usually the *Attacker* argument, while the object is the *Target* argument.

However, although the trigger itself is crucial to determine whether or not there is a reportable event, it is not always sufficient. As a result, most current event extraction systems consider trigger and argument information together to tag a reportable event.

[1] See http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf for a description of this task.

[2] Note that we do not deal with event mention coreference in this paper, so each event mention is treated as a separate event.

[3] An *Entity mention* is a reference (typically, a noun phrase) to an object or a set of objects in one of the semantic categories of interest. A *Timex mention* is a reference to a time expression.

In this paper, we adapted an existing state-of-the-art English IE system [Grishman et al. 2005][4] to serve as our baseline system. This system extracts events independently for each sentence, because the definition of event mention argument constrains them to appear in the same sentence.

In the training process, a pattern collector is first applied to all annotated events to build features used in later classifiers, and then three Maximum Entropy-based classifiers are trained. For each word (trigger candidate) in the training data, if it is a verb, noun or adjective (the three possible parts-of-speech for a trigger in ACE), we update the following three classifiers:

- **Argument Classifier:** Given the trigger candidate and a *Entity / Timex mention* in the sentence, to distinguish whether the mention is a possible argument of a specific event type;
- **Role Classifier:** for each argument identified by the argument classifier, to determine its role with respect to a specific event type;
- **Trigger Classifier:** Given the trigger candidate, the pattern representing the local syntactic context, and a set of roles identified by the role classifier, to determine whether this word is a true trigger, and this is a reportable event.

In the test procedure, for each word (each potential trigger), the argument / role classifier is applied to collect the possible arguments/roles connected to this word, and then the trigger classifier is used to decide whether it is a trigger or not. If it is, an event mention including the trigger and all its roles will be reported, else the word will not be tagged as a trigger, and all the arguments/roles collected by previous classifiers are discarded.

3 Active Learning for Event Extraction

Active learning has been successfully applied to a number of natural language processing tasks, such as named entity recognition (Shen et al. 2004; Hachey, Alex and Becker 2005; Kim et al. 2006), text categorization (Schohn and Cohn 2000; Tong and Koller 2002; Hoi, Jin & Lyu 2006), part of speech tagging (Ringger et al. 2007), parsing (Osborne and Baldridge 2004; Becker and Osborne 2005; Reichart and Rappoport 2007), and word sense disambiguation (Chen et al. 2006; Zhu and Hovy 2007). However, there have not yet been any studies to use active learning in event extraction.

There are several sampling methods in active learning; the most commonly used ones include uncertainty-based sampling, committee-based sampling, and co-testing. Co-testing (Muslea et al. 2000) involves two (or more) redundant views; it simultaneously trains a separate classifier for each view, and the system selects a query based on the degree of disagreement among the learners. Because well-informed classifiers for the two views should agree, co-testing will select an example which is informative for at least one of the classifier models.

In theory, co-testing has some advantages over uncertainty sampling and committee-based sampling. However, the disadvantage of co-testing is that it has more constraints: the two views should be disjoint and each sufficient to learn a classifier. As discussed above, event extraction is complicated and involves several classifiers on different levels interacting together. This makes it difficult to split the feature set into two views. In particular, the identity of the trigger will be a critical feature for any successful classifier. Committee-based sampling faces similar problem as co-testing: it is hard to generate several classifiers that are consistent with the training set or sub-samples of it, respectively.

This leaves uncertainty-based sampling as an attractive option. Although Muslea (2000) points out that uncertainty sampling may make queries that lead to minimal improvements of the classifier, and therefore require more queries to build an accurate classifier, it is simple and can be applied to almost all kinds of statistical models.

We could do active learning at the token level: – asking the *oracle* whether a specific token triggers an event – but that is not very practical. Rather, for each query, we return a sentence that might contain an event to ask the oracle to annotate. We do so because the oracle needs to read the whole sentence to decide whether it is a reportable event, and annotate all its arguments. Thus, a sentence-based sampling pool is built where each sentence is treated as a sample query.

3.1 Applying Uncertainty-based Sampling

Event extraction is a compound classification task, which involves the identification of argu-

[4] The existing system had both pattern matching and statistical components; we integrated these components so that the resulting system would have a uniform probabilistic model suitable for the active learning strategies we employed.

ments/roles, and the event trigger. These classifiers are separately trained, but not independent; results from previous classifiers are used as features for the following classifier, and the decision by the following classifier will affect the previous results (arguments confidently tagged by the argument/role classifier will be discarded if the trigger labeling treats it as not a event). Because the final classifier – the trigger classifier – takes all the considerations we mentioned above as input, and makes a final decision of a reportable event, we use its output as the probability of the event tagger. The traditional approach in uncertainty sampling (Lewis and Gale 1994) queries one of the samples on which the classifier is the least confident. In our case, the greatest uncertainty regarding the presence of an event corresponds to the trigger probability closest to 0.5. We treat the uncertainty of the sentence as the maximum of the uncertainties of the constituent words (i.e., the uncertainty attributable to the word with probability closest to 0.5):

$$e\_Info(S_i) = 1 - \min_{w_j \in S_i} |0.5 - prob\_e(w_j)|$$

where $prob\_e(w_j)$ is the trigger probability of the word w_j in S_i , as returned by the event tagger.

3.2 Problems with Uncertainty-based Sampling

However, the results of uncertainty-based sampling are somewhat disappointing (see Figures 3, 4, and 5 in section 5.3). It performs quite well at first: within a few iterations, trigger labeling (event detection) quickly achieves a performance (F score) of 65%, but beyond that point the gain is very slow. At this point there is still a 7% gap between its performance and that of a classifier trained on the whole sampling pool.

Why does uncertainty-based AL perform this way? The event tagger depends primarily on the particular trigger and secondarily on its local structure, for example, the potential arguments in the immediate vicinity of the trigger and the dependency paths between them. Such information is effective at identifying the trigger and arguments, but is responsive only to particular words and patterns. Triggers and structures which have not been seen in the training data will be assigned uniformly low probabilities. When trained on the whole ACE 2005 corpus (in a supervised training scenario) this is appropriate behavior: we don't want to report an event in testing if we haven't seen the trigger before.[5] However, for active learning, the inability to differentiate among potential new triggers and local structures is critical. Only a few words ever serve as possible triggers for a specific event type. For the *Attack* event, only 2.0% of the words in the ACE training data ever act as an event trigger. The uncertainty of the event tagger, by itself, does not provide useful guidance regarding possible additional triggers the user should be asked about, and the system might query a lot of irrelevant sentences with unseen words before a sentence with a new trigger is found.

We can see this as an instance of a more general problem. Our goal in AL is to select for labeling those data points which are most likely to improve the accuracy of the model. Methods like uncertainty-based sampling are heuristics towards that end, but are not always effective; their success depends on characteristics of the classifier and the feature space. For event extraction, the classifier is most likely to benefit from finding *new*, *frequently-occurring* triggers. We need a way of identifying likely candidates.

Furthermore, we note that – while the final trigger classifier which we train from the labeled data must operate at the token level – we will be presenting the user with a sentence to label, so it is sufficient for the classifier we use for AL to operate at the sentence level.

3.3 Another View from Sentential Scope

Can we find a classifier which suits the needs of our active learner by identifying sentences which are likely to contain an event? A simple (bag-of-words) classifier based on the words in the sentence can do quite well at this task. For example, a sentence with "troops", "victim", "bloody" and "soldier" might be more likely to contain an *Attack* event, even if these words might not be elements of the event.

These bag-of-words features are not particularly helpful for the original task of identifying an event (trigger and arguments) – they don't pinpoint a particular word as the trigger. But that's not a problem if the data selection for AL is operating at a coarser level.[6]

[5] Unlike some other tasks such as named entity and part-of-speech tagging, local contextual clues by themselves are generally not strong enough to reliably tag an event.

[6] Note that some active learners for tasks such as named entities and part-of-speech which also train token-level annotators choose to present data to the user at the sentence level, because it is more convenient and efficient for the user. These taggers could select data at the token level using

3.4 Pseudo Co-Testing

The sentence-level bag-of-words classifier is far from perfect – the predictions at the sentence level are somewhat noisy. But considering that only 6.5% of the sentences in the ACE data contain an *Attack* event, returning a possibly relevant sentence is much more useful than returning a totally irrelevant sentence. If a sentence *S* in the sample pool shares many words with another sentence in the training data known to contain an event, and the event tagger does not find a trigger word in *S*, there is a good chance that *S* contains a new (previously unseen) trigger word and new local structure, because the two sentences may be describing the same event, but using different verbs and word sequences.

Thus, we apply a pseudo co-testing algorithm with one view from an event tagger based on local information, and another view, which aimed to solve an approximate task: whether there is a possible event in a sentence.

We call this algorithm "*pseudo* co-testing" because one of the views is not sufficient to solve the target problem, but is sufficient to solve a subproblem at a coarser granularity, in contrast to traditional co-testing. People might argue that when a *pseudo contention point* is found in this algorithm, it means that at least one of the classifiers is wrong, but we do not know (until we query the oracle) which one. If it is the event tagger, this sample is informative for the event tagger and adding this sample will improve the performance; if it is the sentence classifier, it is not guaranteed that this sample is informative for the event tagger. However, since the updated sentence classifier will serve to select subsequent queries, samples informative for the sentence classifier should accelerate subsequent active learning. Furthermore, the event tagger and the sentence classifier each have their own advantages in finding an event to query. The event tagger prefers sentences with already-known local patterns, like a trigger and its arguments, although the overall sentence (the choice of words and wider structure) might be very different. The sentence classifier prefers sentences sharing the same words, but which may have different local structures. Together they offer the potential for finding new triggers which do not appear in the existing training data (via the sentence classifier) and then acquiring event and non-event exemplars of these triggers (through the event tagger).

In pseudo co-testing, we use the probabilities from the event tagger and sentence classifier to build a contention set consisting of those sentences where the event tagger and sentential event recognizer make different predictions. Among these sentences, we assume that the larger the margin between the event tagger and sentential event recognizer, the less certain the sample is. So, instead of randomly choosing samples from the contention set, we order the samples by their margins between the event tagger and sentential event recognizer, and pick the ones with largest margin:

$$co\_Info(S_i) = \underset{w_j \in S_i \& isCP}{Max} \left| prob\_e(w_j) - prob\_s(S_i) \right|$$

where $prob\_s(S_i)$ is the probability from the sentence classifier; while $prob\_e(w_j)$ is the trigger probability from the event tagger for the word w_j in sentence S_i, and w_j is a *contention point* (CP) where the event tagger's prediction is opposite that of the sentence classifier.

4 Multi-criteria-based AL

Normally active learning only considers the *informativeness* of the sample. In uncertainty-based query, *informativeness* is represented by the least confident sample; in committee-based querying, it is represented by the samples on which the committee vote is the most equally split; in co-testing, it is represented by the contention sample. Shen et al. (2004) pointed out that we should maximize the contribution of the selected instances based on multiple criteria besides *informativeness*. For example, the *representativeness* and *diversity* of the sentence should also be considered. In this way, we not only consider whether the current model contains enough information to classify this sentence (as containing an event), but also consider the distribution of this sample in the whole sampling pool (*representativeness*), and moreover, insure that we select different kind of samples in a batch to make the selection more diverse (*diversity*).

4.1 Features used in Similarity of Samples

To evaluate the *representativeness* and *diversity*, we first need to calculate the similarity between two samples, in our case, two sentences. In general, a sentence will be represented as a vector of features $S_1 = \{f_{11}, f_{12}, f_{13}, ..., f_{1n}\}$ and

two views based on the identity of a token and its immediate context. We share these user considerations, but in addition selecting data at the sentence level enables us to create effective complementary views for event extraction not available at a finer (token) level.

the similarity is calculated based on the feature vectors of the two sentences. Thus, the essential problem becomes how to build the feature vector for a sentence. Since there are two classifiers in the pseudo co-testing, we use features from both classifiers, and measure the similarity using a cosine measure, following Shen et al (2004):

$$Sim(S_1,S_2) = \frac{\sum_{f_i \in S_1} \sum_{f_j \in S_2} sim(f_i,f_j)}{|S_1| \quad |S_2|}$$

where $sim(f_i,f_j)$ is 1 when f_i and f_j are the same, otherwise 0.

4.2 Representativeness

A few prior studies have considered this selection criterion (McCallum and Nigam 1998; Tang et al. 2002; Shen et al. 2004). The *representativeness* of a sample can be evaluated based on how many samples are similar to this sample. Adding samples which are more representative to the training set will have an effect on a larger number of unlabeled samples.

For every sentence in the sampling pool, we measure its *representativeness* based on its average similarity to other sentences in the sampling pool:

$$Represent(S_i) = \frac{\sum_{S_j \in P, i \neq j} sim(S_i,S_j)}{|P|-1}$$

where P is the current sampling pool. In this way, we will filter out the samples that are rare in the whole sampling pool, and focus our effort on the samples that appear more frequently in the whole corpus.

In addition to favoring the most informative example, we also prefer the most representative example. To combine scores from *informativeness* and *representativeness*, we followed Shen et al (2004)'s metric:

$$Score(S_i) = \lambda \cdot co\_Info(S_i) + (1-\lambda) Represent(S_i)$$

where the relative importance of each criterion is determined by the parameter λ ($0 \le \lambda \le 1$). In our experiment, λ is set to 0.7.

4.3 Diversity

The role of the *diversity* criterion is to maximize the training utility of a batch of samples. As we add a batch of samples into the training data in one iteration (for efficiency in updating the model), we want to make sure we provide various types of sentences, which provide the most information as a whole, and avoid selecting very similar sentences for a single batch. To this end, after we rank the sentences in the sampling pool, based on the different strategies mentioned above, we skip over any sentence whose similarity to one already selected in the same batch exceeds a threshold (see Figure 2).

The diversity metric is involved in selecting a batch of instances, as follows:

```
==========================
Given: SenSet = (S1,...,SN) and the BatchSet with the
maximal size K.
Initialization: BatchSet = empty
Loop until BatchSet is full
        Select Si based on some measure from SenSet;
        RepeatFlag = false;
        Loop from j = 1 to CurrentSize of BatchSet
                If Score(Si, Sj) > threshold Then
                        RepeatFlag = true;
                        break;
        If RepeatFlag == false Then
                Add Si into BatchSet.
==========================
```

Figure 2. Diversity criterion in batch-based active learning

5 Experiments

We use the ACE 2005 training corpus, which contains in total 598 annotated documents, to simulate the active learning process. For evaluation, we conduct a blind test on a set of 54 randomly chosen documents. For each active learning strategy, we make 4 runs and use the average scores as our final results. For each run, 10 documents are randomly chosen as the initial training data, and the rest (534 documents) are used to build the sampling pool. Overall, the average initial training set contains 369 sentences, and the sampling pool contains an average of 12074 sentences.

A Maxent model based on bag-of-words features serves as the sentence classifier. To reduce data sparseness, all inflected words are changed to their lemma form (e.g. "attackers"→"attacker"). A list of stop words is also applied.

For each iteration, we picked 50 sample sentences at the top of the ranked list based on different query strategies. To simulate the user queries, annotations extracted from the key annotations are returned as user feedback, and added into the training data.

5.1 Query Strategies

In the following sections, we compare the performance of the query strategies mentioned above – uncertainty-based query (*Uncertainty*), pseudo co-testing (*pCT*), and multi-criteria pseudo co-testing(*multi_pCT*). We employ a random sampling (*Random*) method as a baseline, where samples are selected randomly to add to the training data. Also, to assess the benefit of active learning, we report the performance from the event tagger trained on the entire ACE2005 data except for the test set (*Full_Corpus*).

5.2 Results

The performances (F-measure) of different strategies are evaluated based on three metrics: argument/role labeling (Figures 3 & 4) and trigger labeling (Figure 5).

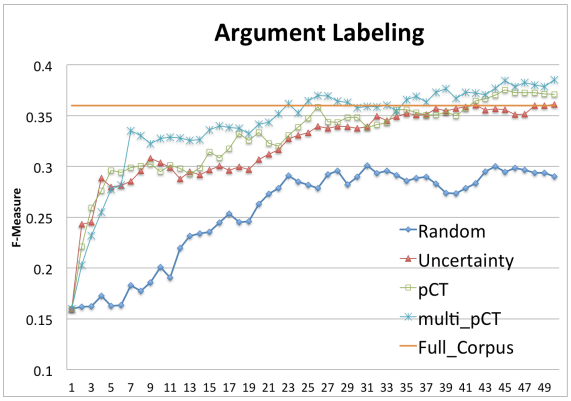

Figure 3. Performance (F-Measure) of argument labeling

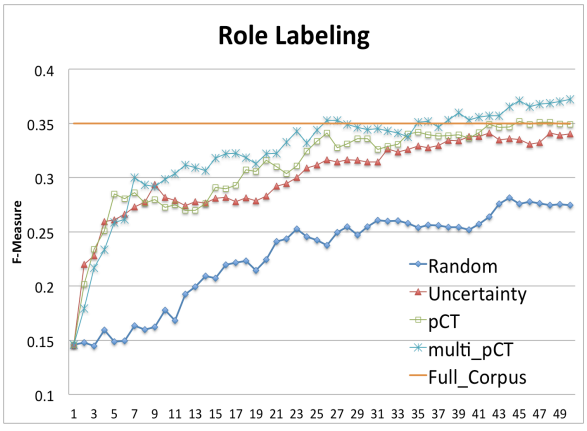

Figure 4. Performance (F-Measure) of role labeling

Uncertainty-based querying (*Uncertainty*) yields poorer results than the other active learning strategies, because of the event tagger's relatively rigid matching procedure. Thus, it lacks the ability to recognize new potential triggers or patterns. For example, if we have pattern *A* which is very similar to some event-bearing patterns in the training data, and pattern *B* which is quite different from any pattern in the training data, the event tagger will treat them the same. However, the sentence classifier provides more graded matching, and gives the sentence containing pattern *A* higher score because they share a lot of words. Thus, the pseudo co-testing (*pCT*) would give a higher score to pattern *A*, and achieve better performance. Also, we observed that multi-criteria pseudo co-testing (*multi_pCT)* performs best in all three evaluations.

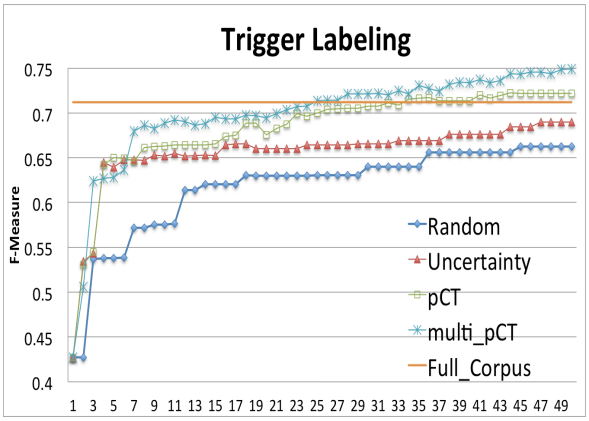

Figure 5. Performance (F-Measure) of trigger labeling

The differences between the approaches are particularly marked for trigger labeling after just a few iterations. Consider how much data must be annotated to get to 95% of full corpus score for trigger labeling (F-Measure 67.5%): *multi_pCT* only takes 7 iterations; *pCT* takes 17 iterations; *Uncertainty* takes 38 iterations. In other words, 5.8%, 9.8%, 18.2% of the whole corpus needs to be annotated to reach the same performance. Thus, using *pCT* is almost twice as fast as *Uncertainty* to reach a reasonable performance, while *multi_pCT* will shorten this process by half again. The benefits of better query selection are clearest for the first few batches of queries, which may be the range of greatest practical import for developers wanting to quickly add new event types.

Overall, we observe that pseudo co-testing performs better on all three evaluation measures than uncertainty-based active learning. Uncertainty-based active learning requires more than 100 iterations before it reaches the level of performance on all three measures achieved by the

supervised system, trained on the entire corpus (*Full_Corpus)*. *pCT* takes 41 iterations to reach this level. At this point, there are in total 369+2050 = 2419 sentences in the training data; this represents a reduction in labor over sequential annotation of 80.6%. Applying the multi-criteria-based strategy (*multi_pCT*), we can reach this point even earlier, in iteration 23, where the labor is reduced by 87.8%[7].

6 Related Work

Many existing active learning methods are based on selecting the most uncertain examples using various measures (Thompson et al. 1999; Schohn and Cohn 2000; Tong and Koller 2000; Engelson and Dagan 1999; Ngai and Yarowsky 2000). (McCallum and Nigam 1998; Tang et al. 2002) proposed methods that consider the representativeness criterion in active learning. (Tang et al. 2002) use the density information to weight the selected examples but do not use it to select a sample. (Brinker 2003) first incorporated diversity in active learning for text classification. Shen et al. (2004) proposed a multi-criteria-based active learning approach and applied it to named entity recognition. They jointly consider multiple criteria, including *informativenes*s, *representativeness* and *diversity*. Experiments showed that incorporating all the criteria together is more efficient than single-criterion-based methods.

Traditional active learning with redundant views splits the feature set into several sub-sets or views, each of which is enough, to some extent, to describe the underlying problem. Muslea et al. (2000) presented an approach in which two classifiers are trained only on labeled data, then run over the unlabeled data. A contention set of examples is then created, consisting of all unlabeled examples on which the classifiers disagree. Samples are randomly selected from this set for query, and then both classifiers are retrained.

To the best of our knowledge, there is no study yet of active learning in event extraction. However, Patwardhan and Riloff (2009) presented a model for role filling in event extraction that jointly considers both the local context around a phrase and the wider sentential context in a probabilistic framework. They used a sentential event recognizer and a plausible role-filler recognizer to jointly make decisions on a sentence, and find the roles of the events. Although it is not a co-testing process, it gave us the intuition of using a sentential view to predict possible events in a sentence.

7 Conclusion

In this paper, we investigate strategies of active learning for event extraction, and propose a novel way of selecting good samples to be added to the training pool. Experiments show that a classifier for a coarser task can provide an extra view to build a pseudo co-testing strategy. Although the ultimate goal involves training the original (fine-grained) classifier, the coarser task can provide useful information for query selection. In the special case of event extraction, we find that a sentence classifier can help an event tagger select a better query, because it is not only good at finding new trigger and local structures from graded matching over a wider scope, but also provides a better way of judge the representativeness and diversity of the samples. In our experiment, we reduced human labor by 80.6% to 87.8%.

Acknowledgments

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

References

Markus Becker and Miles Osborne 2005. A two-stage method for active learning of statistical grammars. *In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence. Edinburgh, Scotland, UK.*

K. Brinker. 2003. Incorporating Diversity in Active Learning with Support Vector Machines. *In Proceedings of ICML, 2003.*

[7] We observe that the AL can perform better than training on the whole corpus; we believe that this is a result of AL selecting more positive training data. After 50 iterations of *multi-pcT*, 31.4% of the selected sentences have positive *Aattack* examples, whereas only 6.1% of the entire corpus has such positive examples. Separate experiments suggest that using a corpus richer in positive examples can produce a small improvement in performance.

Jinying Chen, Andrew Chein, Lyle Ungar and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. *In Proceedings of the HLT-NAACL 2006. New York, USA.*

S. A. Engelson and I. Dagan. 1999. Committee- Based Sample Selection for Probabilistic Classifiers. *Journal of Artifical Intelligence Research.*

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. *In Proc. ACE 2005 Evaluation Workshop, Gaithersburg, MD.*

B. Hachey, B. Alex, and M Becker. 2005. Investigating the effects of selective sampling on the annotation task.. *In proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), 144-151. ACL, Ann Arbor, Michigan, USA.*

Steven C.H Hoi, Rong Jin and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. *Proceedings of the 15th International World Wide Web Conference (WWW 2006). Edinburgh, Scotland.*

R. Jones, R. Ghani, T. Mitchell, and E. Riloff. 2003. Active Learning for Information Extraction with Multiple View Feature Sets, *ECML-03 Workshop on Adaptive Text Extraction and Mining*

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-won Cha, and Gary Geunbae Lee. 2006. MMR-based active machine learning for bio named entity recognition. *In Proceedings of the HLT-NAACL 2006. New York, USA.*

A. McCallum and K. Nigam. 1998. Employing EM in Pool-Based Active Learning for Text Classification. *In Proceedings of ICML, 1998.*

Muslea, I., Minton, S., & Knoblock, C. (2000) Selective sampling with redundant views. *Proc. Of National Conference on Artificial Intelligence (pp. 621-626)*

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. *In Proceedings of Human Language Technology Conference- the North American Chapter of the Association for Computational Linguistics Annual Meeting. (HLT-NAACL 2004). Boston, Massachusetts, USA.*

S. Patwardhan and E. Riloff. 2009. A Unified Model of Phrasal and Sentential Evidence for Information Extraction. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*

Roi Reichart and Ari Rappoport 2007. An ensemble method for selection of high quality parses. *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007).*

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi and Deryle Lonsdale 2007. Active Learning for part-of-speech tagging: Accelerating corpus annotation. *In proceedings of the Linguistic Annotation Workshop. ACL, Prague, Czech Republic. 2007.*

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000). Stanford, California, USA.*

D Shen, J Zhang, J Su, and G Zhou. 2004. Multi-Criteria-based Active Learning for Named Entity Recognition. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, 2004.*

M. Tang, X. Luo and S. Roukos. 2002. Active Learning for Statistical Natural Language Parsing. *In Proceedings of the ACL 2002.*

C. A. Thompson, M. E. Califf and R. J. Mooney. 1999. Active Learning for Natural Language Parsing and Information Extraction. *In Proceedings of ICML 1999.*

Simon Tong and Daphne Koller 2002. Support vector machine active learning with applications to text classification. *Journal of Machine Learning 2 (Match): 45-66*

Jingbo Zhu and Eduard Hovy 2007. Active Learning for word sense disambiguation with methods for addressing the class imbalance problem. *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.*