Automatic Training Data Collection for Semi-Supervised Learning of Information Extraction Systems

Rosie Jones
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
rosie.jones@cs.cmu.edu

Rayid Ghani
Accenture Technology Labs
161 N Clark St, Chicago, IL 60601 USA
rayid.ghani@accenture.com

ABSTRACT

Learning to extract useful information from text data using only minutes of user time means that we need to leverage unlabeled data to accompany the small amount of labeled data. Several algorithms have been proposed to learn from a small amount of labeled data augmented by large amounts of unlabeled data, but no systematic methods exist to collect this unlabeled data. There exists a tension between using randomly selected and therefore very general data, on the one hand, and non-randomly selected but informative unlabeled data on the other hand. In this paper, we describe several methods of automatically collecting large amounts of unlabeled data from the Web to use in semi-supervised Information Extraction systems. These methods are driven by the information extraction task at hand and need only a handful of seed examples. We show that when the target class is very sparse, non-random data selected to contain multiple instances of known positive examples can help learn patterns for information extraction from text data. This result holds even when the class frequency is misrepresented in automatically collected data. We also present results obtained by interleaving our approach with semi-supervised algorithms for information extraction, and find that our training data collection approach outperforms data collected uniformly at random from the web, and performs competitively with the standard approach normally used in studies of semi-supervised learners where the unlabeled set is fixed and pre-collected independent of the task.

1. INTRODUCTION

Information Extraction systems offer a way of automating the discovery of information from text documents. Both research and commercial systems for information extraction currently need large amounts of labeled training data to learn dictionaries and patterns to use for extraction. Collecting these large numbers of labeled examples can be very expensive, thus emphasizing the need for algorithms that can provide accurate classifications after labeling only a few examples. One way to reduce the amount of labeled data required is to develop algorithms that can learn effectively from a small number of labeled examples augmented with a large number of unlabeled examples.

Several algorithms have recently been proposed for bootstrapping from very few examples for several text learning tasks. Metabootstrapping [1] is an approach to learning dictionaries for information extraction starting only from a handful of phrases which are examples of the target class. It makes use of the fact that noun-phrases and the partial-sentences they are embedded in can be used as two complementary sources of information about semantic classes. Similar methods have been used for named entity classification [2].

More recently, Yangarber et al. used seed information extraction template patterns to find target sentences from unlabeled documents, then assumed strongly correlated patterns are also relevant, for learning new templates. They used an unlabeled corpus of 5,000 to 10,000 documents, and suggest extending the size of the corpus used, as many initial patterns are very infrequently occurring [3, 4].

The field of text categorization has also seen several algorithms proposed for combining labeled and unlabeled data. Expectation Maximization to estimate maximum a posteriori parameters of a generative model for text classification [5], constructing generative models from unlabeled data to perform discriminative classification [6], transductive inference for support vector machines to optimize performance on a specific test set [7] are some examples that have shown that using unlabeled data can significantly improve classification performance, especially when labeled training data are sparse. A related set of research uses labeled and unlabeled data in problem domains where the features naturally divide into two disjoint sets. Blum and Mitchell [8] present an algorithm for classifying web pages that builds two classifiers: one over the words that appear on the page, and another over the words appearing in hyperlinks pointing to that page. Data sets whose features naturally partition into two sets, and algorithms that use this division, fall into the co-training setting [8].

The document classification work described above, as well as recent research in selective sampling [9] and active learning [10] uses data which was originally collected *and labeled* for the task at hand, then removes the labels to test the algorithm's use of unlabeled data. The same is true for most experimental work in semi-supervised learning [11, 12, 13,

CIKM 2002 McLean, VA USA

14]. This methodology of *hiding* the labels and pretending that part of the data that was initially labeled is now unlabeled, in order to test the performance of semi-supervised learning algorithms, has shown that unlabeled data can be useful for learning. However, it has not provided systematic methods for actually collecting the unlabeled data that can be used in real-world systems.

A major reason for not retrieving pages uniformly at random from the web to use as unlabeled training data is that machine learning algorithms traditionally assume that training and test data come from the same distribution. Randomly picking a training set that neither matches the distribution of the initial labeled examples nor of the test set, would severely violate this assumption and can potentially hurt the performance of the classifier. Could the secret of the success of these bootstrapping algorithms be the fact that the unlabeled data is coming from *exactly* the same distribution as the test set? Would these algorithms fail to benefit from unlabeled data that might not follow the distribution of the test set? Can these algorithms still be used in a practical setting where only a limited amount of labeled data (and no pre-collected set of unlabeled data) is available?

These questions lead us to investigate methods that target practical information extraction tasks where large amounts of training data are not available. We propose data collection methods that can be seeded from a small set of labeled training data, and adaptively construct Web search queries in order to collect data that is truly unlabeled and is suitable for the information extraction task at hand. In particular, we describe a number of methods for collecting training data for information extraction tasks, based on knowledge of the target class from a few training examples. In order to analyze the effectiveness of these methods, the data collected is then used as training data for semi-supervised learning algorithms for information extraction, such as meta-bootstrapping [1] and co-EM [14]. The Web query generation approach is motivated by our previous work on adaptive query generation to find documents matching minority concepts on the Web [15]. In this work, we consider the semantic class to be extracted as the concept and use web queries to find documents containing high densities of that concept. We also provide a taxonomy of the ways class-knowledge can be used to aid data collection, and describe the experiments we use to provide empirical proof of the efficacy of these.

2. ALGORITHMS FOR BOOTSTRAPPING INFORMATION EXTRACTION

In this section we describe two baseline semi-supervised machine learning algorithms to train information extraction systems: metabootstrapping and coEMIE.

2.1 Metabootstrapping

Meta-bootstrapping [1] is a simple two-level bootstrapping algorithm using two features sets to label one another in alternation. It is customized for information extraction, using the feature sets *noun-phrases* and *noun-phrase-contexts* (or *caseframes*). There is no notion of negative examples or features, but only positive features and unlabeled features. The two feature sets are used asymmetrically. The noun-phrases are used as initial data and the set of positive features grows as the algorithm runs, while the noun-phrase-contexts are relearned with each outer iteration.

Heuristics are used to score the features from one set at each iteration, based on co-occurrence frequency with positive and unlabeled features, using both frequency of co-occurrence, and diversity of co-occurring features. The highest scoring features are added to the positive feature list.

2.2 coEMIE

Co-EM, is a hybrid algorithm, originally proposed by Nigam & Ghani [14] for text categorization, combining features from both co-training and Expectation Maximization (EM). Both of these algorithms have been used earlier for semi-supervised text classification. Co-EM is iterative, like EM, but uses the feature split present in the data, like co-training. It uses a feature split in the data and builds one classifier for each feature set. It proceeds by initializing the first-feature-set classifier (classifier A) from the labeled data only. Then, A probabilistically labels all the unlabeled data. The second-feature-set classifier (B) then trains using the labeled data and the unlabeled data with A's labels. B then relabels the data for use by A, and this process iterates until the classifiers converge. A and B predictions are combined together as co-training embedded classifiers are.

Muslea et al. [9] extend the co-EM algorithm to incorporate active learning and show that it has a robust behavior on a large spectrum of problems because of its ability to ask for the labels of the most ambiguous examples, which compensates for the weaknesses of the underlying semi-supervised algorithm.

We use coEMIE, which is an implementation of co-EM for semi-supervised information extraction. The feature set is the same as that used for meta-bootstrapping, and consists of noun-phrases (set A), and their local contexts (set B).

3. CORPUSBUILDER

In this section we give some background on CorpusBuilder [15]; originally designed for collecting corpora in a minority language on the Web. In the following section we describe how we adapt this set-up to collect data for training information extractors.

CorpusBuilder iteratively creates new queries, in order to build a collection of documents matching a concept. The target concept is defined by one or more initial documents provided by the user, and a concept classifier (text classifier). At a high level, CorpusBuilder works by taking as initial input from the user two sets of documents, relevant and non-relevant. Given these documents, it uses a term selection method to select words from the relevant and non-relevant documents to be used as inclusion and exclusion terms for the query, respectively. This query is sent to the search engine and the highest ranking document is retrieved, passed through the classifier and added to the set of relevant or non-relevant documents according to the classification. The process is then iterated, updating the set of documents that the words are selected from at each step.

Ghani et al. [15] compared several different metrics to select query words at each iteration and found that using Odds-Ratio to score words and then picking the top n highest scored words performed consistently well. In this paper, we will therefore only use Odds-Ratio and Weighted Odds-Ratio scores to select query words when using CorpusBuilder. We also implemented a method which selects new query terms by using one of the bootstrapping algorithms to find good terms from newly collected documents.

4. TRAINING DATA COLLECTION

We now describe a number of methods for collecting *unlabeled* training data, based on knowledge of the target class from a few training examples (seed words). We provide a taxonomy of the ways class-knowledge can be used to aid data collection, and describe the experiments we use to provide empirical proof of the efficacy of these. We also provide a justification for using non-random data collected once we have seen training examples of an extremely sparse class.

4.1 Standard Approach

We describe here the standard approach used in other studies with semi-supervised learning algorithms for text learning. The general approach is as follows:

1. Start with a set of labeled training data.
2. Hide the labels of a percentage of the training data (the *fixed-unlabeled set*) and pretend it is unlabeled.
3. Initialize the semi-supervised learner with the rest of the training data (seed examples).
4. Build the model using the labeled data and the fixed-unlabeled set.
5. Apply the learned model to a fixed test set and evaluate.

We implement this approach mainly for comparing our methods with the standard approach that is used in related literature, and refer to it as *fixed* in our results. It is important to note that this approach is not applicable for arbitrary information extraction tasks since a labeled training corpus is usually not available. In some ways, the performance of the information extraction algorithm on the *fixed* (manually collected) data serves as an upper bound. We would expect our proposed methods for automatically collecting the data to be significantly cheaper (in terms of labeling cost) but probably not as effective.

4.2 Random Data Collection

A baseline approach to automatic data collection retrieves a set of pages according to a uniform random distribution from the web, to use as unlabeled data to augment the training data provided to the semi-supervised learner. We use the random link feature available from http://random.yahoo.com/bin/ryl, that displays a random page every time it is loaded. Repeatedly downloading pages from this site, we collect the unlabeled set and then train the semi-supervised learning algorithm. Since this is a uniform sample from the web, there is no reason to expect that there will be many instances of the concept that defines our task, but this method serves as a naive baseline approach.

4.3 Task-specific Data Collection

Machine learning algorithms assume that the training and test collections come from the same distribution. Violating this assumption may lead to high error rates, since error minimization is performed with respect to the distribution on the training data. However, for a very sparse target class, a training set from the same distribution as the test set may contain very few examples of the target class. When we seek to learn locations, for example, there may be very few examples of locations in typical data. When the variety of possible attributes of examples is also very high, there may be very few example in a typical training set with which to learn good representations for the features. For information extraction tasks, for example, examples in the target class may make up less than 10% of data. When we are beginning with seed examples only, the initial density may be very low indeed. The set of possible noun-phrases and contexts that could occur with target examples is very large. Thus if we have bounds on the size of the data set we can use for training (because of disk space, CPU time, or bandwith limitations) it may be most effective to increase the density of the target class in the training set, to give us most effective information for modeling the target class.

4.3.1 Seed-based Data Collection

In seed-based data collection, documents are collected based on the seed-words which define the information extraction task. These seed words are provided by the user to the information extraction system and serve as initial labeled data. We use these seed words to create queries which are then executed in AltaVista and store the retrieved documents which are then used as the unlabeled training set. We can collect documents containing seed words using the following methods:

- *all*: Query for documents that contain all seed words
- *some*: Query for documents that contain k seedwords
- *one*: Query for documents that contain one seedword

In our experiments we used only the *one* method, cycling through all keywords, performing one query for each seedword. This method is referred to as seed-based in the results section.

4.3.2 CorpusBuilder-Augmented Seed-based Data Collection

This method is similar to the seed-based data collection approach where we start with collecting documents that contain the seedwords using AltaVista. We then use the initial set as input to CorpusBuilder to find more documents 'like' the ones that are collected by the seed-based approach. The algorithm proceeds as follows:

1. Given initial seedwords, do an AltaVista query on each of the seed words to find documents that contain those words.
2. Collect a set of negative examples by retrieving a set of general documents that do not contain the seed words. We do this by constructing queries that return documents containing stopwords (this, is, are, of) and not containing any of the seed words.
3. Seed CorpusBuilder with the documents from step 1 as positive examples and those from step 2 as negative examples. The goal of CorpusBuilder is now to find more unlabeled training data by constructing queries that can retrieve documents matching the positive examples.

4.3.3 Interleaved Data Collection

In interleaved data collection, we collect data both before starting the semi-supervised learning, and also when we have acquired more information about the task and the data. This method uses the intuition that the unlabeled data may tell us more about the task, and that this information may be useful for collection further training data. At a high level, we can seed the learning algorithm with documents collected using initial seedwords and after the learning algorithm has learned about the task, we can derive the words/concepts that were informative for the learner and now augment our list of initial seedwords with these words and iterate. This type of methodology provide us with the following methods of constructing queries:

- noun-phrase-based data collection: In this approach, new noun-phrases are learned, and then used as the basis for further data collection
- phrase-context-based data collection: contexts (or general information extraction patterns) are learned, then used as query terms for gathering more data
- combined data collection: both noun-phrases and phrase-contexts are used as query terms for data collection.

In the experiments described in this paper, we performed iterated interleaved data collection, acquiring new data repeatedly, using only noun-phrases as

The details of the algorithm for interleaved data collection are:

1. Initialize with seed words (pos only)
2. select a seed word with uniform probability
3. fetch docs containing seed words (newdocs)
4. run bootstrapping on the newdocs
5. pick new seed words from the newdocs, add to seed word list

return to 2.

This method is referred to as `interleaved` in the Results section.

4.3.4 Selective Data Collection

Selective data collection adds a user into the loop, filtering data acquired to confirm that it is representative of the task. This approach is more expensive, but may lead to better results. We did not perform experiments using this approach but will pursue this promising approach in future work.

5. TEXT PREPARATION AND INFORMATION EXTRACTION TASKS

In this section we describe the data we use as input to the algorithms, the parsing we perform on the data as a preprocessing step, and the information extraction tasks we are training for.

5.1 Text Data

For comparison with previous work, we perform our experiments on the web pages from companies used by Riloff and Jones [1]; a subset of the Industry Sector data set commonly used for text classification. There were 4582 documents, which we divided at random into 4160 documents constituting the *fixed-unlabeled* data, 233 documents constituting the *test* data, and 189 documents for validation, which we did not use in these experiments.

Task	Seed words
Locations	australia canada china england france germany japan mexico switzerland united states
Products and Services	product products service services

Table 1: Seed words used for CorpusBuilder and Information Extraction Bootstrapping

5.2 Text Parsing

Autoslog [16] [17] is a system for parsing and sorting phrases for information extraction, freely available for research purposes. We used Autoslog to parse all documents, and identify all noun phrases NPs and the contexts they were embedded in. This gives us pairs of the form $< located\_in, australia >$ and $< sold, oil\_and\_gas >$". All our data for bootstrapping was in the form of these noun phrase-context pairs. All letters were converted to lower case.

5.3 Information Extraction Tasks

We used two different information extraction tasks for these experiments. We can rank these tasks from easy to difficult, according to whether the target concept can readily be recognized from noun-phrases along (such as locations) to those which require more context to be definitively identified (such as products). The seeds we used to define the task are given in Table 1.

6. EVALUATION

We evaluate our training data acquisition methods in two ways: precision on the training set, and precision and recall for extraction on a held-out test set.

6.1 Precision on the Training Corpus

Since one of the main objectives of an information extraction system is to obtain many examples of the target semantic class, we inspect the noun phrases produced by the bootstrapping algorithms from the training corpus. This training corpus is obtained either randomly, or by automatic construction using our proposed methods from the earlier sections. We take the top 250 noun-phrases produced, and manually evaluate them for precision. Ideally, we would like to perform this manual evaluation at different levels of recall, but due to time restrictions, we only evaluate precision at 250 for each information extraction task used in this paper.

6.2 Extraction on the Test Corpus

Since we may wish to perform information extraction on a specific test collection unseen at the time of corpus construction, we also performed evaluation on an unseen test corpus. This test corpus was preprocessed and all noun phrases were manually labeled as product, location, or neither.

For evaluation on the held-out test set, we trained coEM to convergence output all learned noun-phrases and contexts for each task, together with the learned estimate of probability of the target class. We then assigned a score to each example in the test set by assigning it $P(NP)xP(context)$. We then sort the examples by score and calculate precision and recall with respect to the target class.

7. RESULTS

7.1 Precision on the Training Corpus

Table 2 shows the precision when manually evaluating the top 250 *unique* noun-phrases produced by running bootstrapping on the corpus collected for *products* using the data collection methods described earlier. Some of the methods resulted in fewer than 250 unique NPs, hence the denominator in some of the cells in tables 2 and 3 is less than 250. By observing these tables, we see that by only considering the lexicon produced, we get a greater proportion of target noun-phrases with the seed-word based interleaved method (SWI), than with the fixed corpus in the case of *products*. For *locations*, the fixed corpus performs better. The good performance using the *fixed* corpus is not surprising as the corpus was manually collected specifically for this task; rather, the competitive results obtained using the seed-word based automatic methods are extremely encouraging as it provides validation that the automatic data collection methods are effective and that semi-supervised algorithms can indeed be used for arbitrary information tasks.

Since these results suggest that our automatically collected corpora can be useful in some instances for the information extraction algorithms, we believe that by taking the noun-phrases produced with bootstrapping on SWI may be helpful to augment our seed list when bootstrapping on a test corpus. Again, it is important to note that the *test corpus* we are using in our experiments was initially part of the *fixed corpus* and thus follows a similar distribution. It is very likely that a system trained on the *fixed* corpus would do well on the *test* corpus. Therefore, we do not expect our autmatically collected corpora to outperform the system trained on the *fixed* corpus; rather, we expect it to produce competitive results (and at least, better than a random corpus) since a *fixed* corpus is not available for arbitrary information extraction tasks.

7.2 Extraction on the Test Corpus

Figure 1 shows precision plotted against recall for *locations*, using noun-phrases and caseframes derived from a training corpus, to extract from a held-out test corpus. Highest precision is obtained using interleaved seed-based collection (SWI) , a corpus collected using initial seeds, and seeds acquired during training, outperforming a randomly collected corpus. The fixed corpus, from the same distribution as the test corpus, performs well in the high-recall range. These trends also hold for the task *products*, as shown in Figure 2.

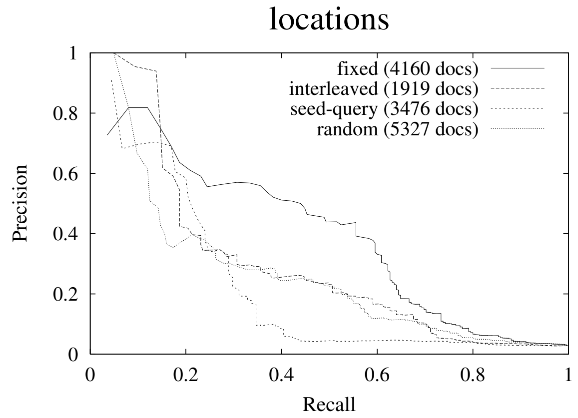

Figure 1: Precision Recall Graph for *locations*, using noun-phrases and caseframes derived using coEMIE from a training corpus, to extract from a held-out test corpus. Highest precision is obtained using interleaved (SWI), a corpus collected specifically to contain examples of the seed-words, outperforming a randomly collected corpus. The fixed corpus, from the same distribution as the test corpus, performs well in the high-recall range.

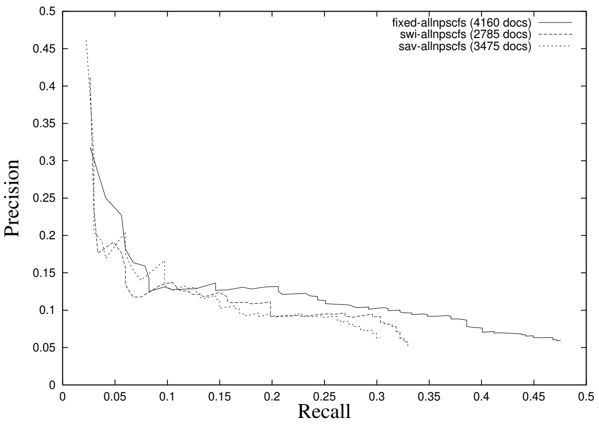

Figure 2: Precision Recall Graph for Products, using noun-phrases and caseframes derived using coEMIE from a training corpus, to extract from a held-out test corpus. Highest precision is obtained using seed-based (SAV), a corpus collected specifically to contain examples of the seed-words. The fixed corpus, from the same distribution as the test corpus, performs well in the high-recall range.

8. SEED-WORD DENSITY

We claimed in Section 4.3 that a greater density of examples of the target concept in the training data would lead to improved learning. We test this hypothesis by comparing seed density in the corpora to the corresponding extraction performance. We calculate seed density by using the seeds in Table 1, and the parsed data for each corpus. We find the number of occurrences of one of the seed-words as a noun-phrase, and divide it by the total number of noun-phrases. Table 4 shows the seed density for the fixed (manually collected), random, seed-based and interleaved corpora,

Corpus (Number of Documents)	Bootstrapping Algorithm	Precision
Fixed (4160)	meta-bootstrapping	72/241 (30%)
Fixed (4160)	coEM	117/256 (46%)
Seed-based with Corpusbuilder (3475)	meta-bootstrapping	62/178 (35%)
Seed-based with Corpusbuilder (3475)	coEM	77/252 (31%)
Seed-based Interleaved SWI (2785)	meta-bootstrapping	71/168 (42%)
Seed-based Interleaved SWI (2785)	coEM	118/256 (46%)

Table 2: Precision at 250: Top 250 noun-phrases for *products*, using sundance 2.41

Corpus (Number of Documents)	Bootstrapping Algorithm	Precision
Fixed (4160)	meta-bootstrapping	175/250 (70%)
Fixed (4160)	coEM	129/262 (49%)
Seed-based with Corpusbuilder (3476)	meta-bootstrapping	158/250 (63%)
Seed-based with Corpusbuilder (3476)	coEM	46/262 (18%)
Seed-based Interleaved SWI (1919)	meta-bootstrapping	105/247 (43%)
Seed-based Interleaved SWI (1919)	coEM	73/262 (28%)

Table 3: Precision at 250: Top 250 noun-phrases for *locations*, using sundance 2.41

Corpus (Num pages; num NPs)	Class	Seed-density (num seeds)
fixed (4161; 266114)	locations	0.0018 (487)
random (755; 35325)	locations	0.0021 (74)
seed-word (3476; 293487)	locations	0.0165 (4849)
fixed (4161; 266114)	products	0.0062 (1650)
random (755; 35325)	products	0.0021 (74)
seed-word (3475; 212542)	products	0.0116 (2468)

Table 4: Seed density for the different corpora

for locations and products. We see that the seed-based data collection leads to a much higher density of seed-words, for both locations and products. The *random* and *fixed* corpora have similar seed densities, though test results differed for these classes. This suggests that seed-density alone is not a good predictor of the suitability of a corpus for bootstrapping using seeds.

9. CONCLUSIONS AND FUTURE WORK

We described a taxonomy of approaches for training data collection that is task-based and uses a small set of seed words as a definition of the task. The methods described collected unlabeled training examples for semi-supervised algorithms containing relatively high numbers of initial seed-words. The task-based automatic methods outperform the extraction performance of a system trained on a random corpus. Furthermore, the performance of the proposed methods was encouraging when evaluated on the unlabeled set itself. Although using the seed words generated by training on the unlabeled corpus to bootstrap extraction on the test corpus was not very effective compared to using the manually collected *fixed* corpus, the result does not seem surprising. The methods described in this paper provide a general framework for designing automatic data collection methods that will extend the applicability of semi-supervised information extraction to arbitrary tasks. Our results here use relatively small data-sets, though the approach lends itself to collecting large unlabeled sets. In future work to plan to extend this to automatic collection of more extensive data collections, as well as adding a user into the loop when constructing the corpus, and an active learning framework to minimize the burden on that user. We also plan to extend this data collection framework to incorporate text classification tasks.

10. REFERENCES

[1] Riloff, E., Jones, R.: Learning Dictionaries for Information Extraction by Multi-level Boot-strapping. In: AAAI-99. (1999)

[2] Collins, M., Singer, Y.: Unsupervised models for named entity classification. In: EMNLP/VLC-99. (1999)

[3] Yangarber, R., Grishman, R., Tapanainen, P., Huttunen, S.: Automatic acquisition of domain knowledge for information extraction. In: COLING-2000. (2000)

[4] Yangarber, R., Grishman, R., Tapanainen, P., Huttunen, S.: Unsupervised discovery of scenario-level patterns for information extraction. In: ANLP-NAACL 2000. (2000) 282–289

[5] Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning **39** (2000) 103–134

[6] Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Advances in NIPS 11. (1999)

[7] Joachims, T.: Transductive inference for text classification using support vector machines. In: Proceedings of ICML '99. (1999)

[8] Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT-98. (1998)

[9] Muslea, I., Minton, S., Knoblock, C.A.: Selective sampling with redundant views. In: AAAI/IAAI. (2000) 621–626

[10] Cohn, D., Ghahramani, Z., Jordan, M.: Active learning with statistical models. JAIR **4** (1996) 129–145

[11] Bruce, R.: Semi-supervised learning using prior probabilities and em. In: IJCAI-01 Workshop on Text Learning: Beyond Supervision. (2001)

[12] Muslea, I., Minton, S., Knoblock, C.A.: Selective sampling + semi-supervised learning = robust multi-view learning. In: IJCAI-01 Workshop on Text Learning: Beyond Supervision. (2001)

[13] Goldman, S., Zhou, Y.: Enhancing supervised learning with unlabeled data. In: ICML. (2000)

[14] Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of co-training. In: CIKM. (2000) 86–93

[15] Ghani, R., Jones, R., Mladenic, D.: Using the web to create minority language corpora. In: CIKM-2001. (2001)

[16] Riloff, E.: Automatically constructing a dictionary for information extraction tasks. In: National Conference on Artificial Intelligence. (1993)

[17] Riloff, E.: An empirical study of automated dictionary construction for information extraction in three domains. Artificial Intelligence **85** (1996) 101–134