Automatically Building Training Examples for Entity Extraction

Marco Pennacchiotti
Yahoo! Labs
Sunnyvale, CA, USA
pennac@yahoo-inc.com

Patrick Pantel
Microsoft Research
Redmond, WA, USA
ppantel@microsoft.com

Abstract

In this paper we present methods for automatically acquiring training examples for the task of entity extraction. Experimental evidence show that: (1) our methods compete with a current heavily supervised state-of-the-art system, within 0.04 absolute mean average precision; and (2) our model significantly outperforms other supervised and unsupervised baselines by between 0.15 and 0.30 in absolute mean average precision.

1 Introduction

Entity extraction is a fundamental task in NLP and related applications. It is broadly defined as the task of extracting entities of a given semantic class from texts (e.g., lists of actors, musicians, cities). Search engines such as Bing, Yahoo, and Google collect large sets of entities to better interpret queries (Tan and Peng, 2006), to improve query suggestions (Cao et al., 2008) and to understand query intents (Hu et al., 2009). In response, automated techniques for entity extraction have been proposed (Paşca, 2007; Wang and Cohen, 2008; Chaudhuri et al., 2009; Pantel et al., 2009).

There is mounting evidence that combining knowledge sources and information extraction systems yield significant improvements over applying each in isolation (Paşca et al., 2006; Mirkin et al., 2006). This intuition is explored by the Ensemble Semantics (ES) framework proposed by Pennacchiotti and Pantel (2009), which outperforms previous state-of-the-art systems. A severe limitation of this type of extraction system is its reliance on editorial judgments for building large training sets for each semantic class to be extracted. This is particularly troublesome for applications such as web search that require large numbers of semantic classes in order to have a sufficient coverage of facts and objects (Tan and Peng, 2006). Hand-crafting training sets across international markets is often infeasible. In an exploratory study we estimated that a pool of editors would need roughly 300 working days to complete a basic set of 100 English classes using the ES framework. Critically needed are methods for automatically building training sets that preserve the extraction quality.

In this paper, we propose simple and intuitively appealing solutions to automatically build training sets. Positive and negative training sets for a target semantic class are acquired by leveraging: i) 'trusted' sources such as structured databases (e.g., IMDB or Wikipedia for acquiring a list of $Actors$); ii) automatically constructed semantic lexicons; and iii) instances of semantic classes other than the target class. Our models focus on extracting training sets that are large, balanced, and representative of the unlabeled data. These models can be used in any extraction setting, where 'trusted' sources of knowledge are available: Today, the popularity of structured and semi-structured sources such as Wikipedia and internet databases, makes this approach widely applicable. As an example, in this paper we show that our methods can be successfully adapted and used in the ES framework. This gives us the possibility to test the methods on a large-scale entity extraction task. We replace the manually built training data in the the ES model with the training data built

Proceedings of the Fifteenth Conference on Computational Natural Language Learning, pages 163–171,
Portland, Oregon, USA, 23–24 June 2011. ©2011 Association for Computational Linguistics

by our algorithms. We show by means of a large empirical study that our algorithms perform nearly as good as the fully supervised ES model, within 4% in absolute mean average precision. Further, we compare the performance of our method against both Paşca et al. (2006) and Mirkin et al. (2006), showing 17% and 15% improvements in absolute mean average precision, respectively.

The main contributions of this paper are:

- We propose several general methods for automatically acquiring labeled training data; we show that they can be used in a large-scale extraction framework, namely ES; and
- We show empirical evidence on a large-scale entity extraction task that our system using automatically labeled training data performs nearly as well as the fully-supervised ES model, and that it significantly outperforms state-of-the-art systems.

2 Automatic Acquisition of Training Data

Supervised machine learning algorithms require training data that is: (1) balanced and large enough to correctly model the problem at hand (Kubat and Matwin, 1997; Japkowicz and Stephen, 2002); and (2) representative of the unlabeled data to decode, i.e., training and unlabeled instances should be ideally drawn from the same distribution (Blumer et al., 1989; Blum and Langley, 1997). If these two properties are not met, various learning problems, such as overfitting, can drastically impair predictive accuracy. To address the above properties, a common approach is to select a subset of the unlabeled data (i.e., the instances to be decoded), and manually label them to build the training set.

In this section we propose methods to automate this task by leveraging the multitude of structured knowledge bases available on the Web.

Formally, given a *target class* c, our goal is to implement methods to automatically build a training set $T(c)$, composed of both positive and negative examples, respectively $P(c)$ and $N(c)$; and to apply $T(c)$ to classify (or rank) a set of *unlabeled data* $U(c)$, by using a learning algorithm. For example, in entity extraction, given the class $Actors$, we might have $P(c) = \{Brad\ Pitt,\ Robert\ De\ Niro\}$ and $N(c) = \{Serena\ Williams,\ Rome,\ Robert\ Demiro\}$.

Below, we define the components of a typical knowledge acquisition system as in the ES framework, where our methods can be applied :

Sources. Textual repositories of information, either structured (e.g., Freebase), semi-structured (e.g., HTML tables) or unstructured (e.g., a webcrawl). Information sources serve as inputs to the extraction system, either for the Knowledge Extractors to generate candidate instances, or for the Feature Generators to generate features (see below).

Knowledge Extractors (KE). Algorithms responsible for extracting candidate instances such as entities or facts. Extractors fall into two categories: trusted and untrusted. *Trusted extractors* execute on structured sources where the contents are deemed to be highly accurate. *Untrusted extractors* execute on unstructured or semi-structured sources and generally generate high coverage but noisy knowledge.

Feature Generators. Methods that extract evidence (features) of knowledge in order to decide which extracted candidate instances are correct.

Ranker. A module for ranking the extracted instances using the features generated by the feature generators. In supervised ML-based rankers, labeled training instances are required to train the model. Our goal here is to automatically label training instances thus avoiding the editorial costs.

2.1 Acquiring Positive Examples

Trusted positives: Candidate instances for a class c that are extracted by a trusted Knowledge Extractor (e.g., a wrapper induction system over IMDB), tend to be mostly positive examples. A basic approach to acquiring a set of positive examples is then to sample from the unlabeled set $U(c)$ as follows:

$$P(c) = \{i \in U(c) : (\exists\, KE_i | KE_i\ is\ trusted\} \quad (1)$$

where KE_i is a knowledge extractor that extracted instance i.

The main advantage of this method is that $P(c)$ is guaranteed to be highly accurate, i.e., most instances are true positives. On the downside, instances in $P(c)$ are not necessarily representative of the untrusted KEs. This can highly impact the performance of the learning algorithm, which could overfit the training data on properties that are specific to the trusted KEs, but that are not representative of the true population to be decoded (which is largely coming from untrusted KEs).

We therefore enforce that the instances in $P(c)$ are extracted not only from a trusted KE, but also from any of the untrusted extractors:

$$P(c) = \{i \in U(c) : \exists\, KE_i | KE_i \; is\ trusted \wedge \\ \exists\, KE_j | KE_j \; is\ untrusted\} \tag{2}$$

External positives: This method selects the set of positive examples $P(c)$ from an external repository, such as an ontology, a database, or an automatically harvested source. The main advantage of this method is that such resources are widely available for many knowledge extraction tasks. Yet, the risk is that $P(c)$ is not representative of the unlabeled instances $U(c)$, as they are drawn from different populations.

2.1.1 Acquiring Negative Examples

Acquiring negative training examples is a much more daunting task (Fagni and Sebastiani, 2007). The main challenge is to select a set which is a good representative of the unlabeled negatives in $U(c)$. Various strategies can be adopted, ranging from selecting near-miss examples to acquiring generic ones, each having its own pros and cons. Below we propose our methods, some building on previous work described in Section 5.

Near-class negatives: This method selects $N(c)$ from the population $U(C)$ of the set of classes C which are semantically similar to c. For example, in entity extraction, the classes $Athletes$, $Directors$ and $Musicians$ are semantically similar to the class $Actors$, while $Manufacturers$ and $Products$ are dissimilar. Similar classes allow us to select negative examples that are semantic near-misses for the class c. The hypothesis is the following:

> *A positive instance extracted for a class similar to the target class c, is likely to be a near-miss incorrect instance for c.*

To model this hypothesis, we acquire $N(c)$ from the set of instances having the following two restrictions:

1. The instance is most likely correct for C
2. The instance is most likely incorrect for c

Note that restriction (1) alone is not sufficient, as an instance of C can be at the same time also instance of c. For example, given the target class *Actors*, the instance *'Woody Allen'* $\in$ *Directors*, is not a good negative example for *Actors*, since Woody Allen is both a director and an actor.

In order to enforce restriction (1), we select only instances that have been extracted by a trusted KE of C, i.e., the confidence of them being positive is very high. To enforce (2), we select instances that have never been extracted by any KE of c. More formally, we define $N(c)$ as follows:

$$N(c) = \bigcup_{c_i \in C} P(c_i) \setminus U(c) \tag{3}$$

The main advantage of this method is that it acquires negatives that are semantic near-misses of the target class, thus allowing the learning algorithm to focus on these borderline cases (Fagni and Sebastiani, 2007). This is a very important property, as most incorrect instances extracted by unsupervised KEs are indeed semantic near-misses. On the downside, the extracted examples are not representative of the negative examples of the target class c, since they are drawn from two different distributions.

Generic negatives: This method selects $N(c)$ from the population $U(C)$ of all classes C different from the target class c, i.e., both classes semantically similar and dissimilar to c. The method is very similar to the one above, apart from the selection of C, which now includes any class different from c. The underlying hypothesis is the following:

> *A positive instance extracted for a class different from the target class c, is likely to be an incorrect instance for c.*

This method acquires negatives that are both semantic near-misses and far-misses of the target class. The learning algorithm is then able to focus both on borderline cases and on clear-cut incorrect cases, i.e. the hypothesis space is potentially larger than for the near-class method. On the downside, the distribution of c and C are very different. By enlarging the potential hypothesis space, the risk is then again to capture hypotheses that overfit the training data on properties which are not representative of the true population to be decoded.

Same-class negatives: This method selects the set of negative examples $N(c)$ from the population $U(c)$. The driving hypothesis is the following:

> *If a candidate instance for a class c has been extracted by only one KE and this KE is untrusted, then the instance is likely to be incorrect, i.e., a negative example for c.*

The above hypothesis stems from an intuitive observation common to many ensemble-based paradigms (e.g., ensemble learning in Machine Learning): the more evidence you have of a given fact, the higher is the probability of it being actually true. In our case, the fact that an instance has been extracted by only one untrusted KE, provides weak evidence that the instance is correct. $N(c)$ is defined as follows:

$$N(c) = \{i \in U(c) : \exists!\ KE_i\ \wedge KE_i\ is\ untrusted\} \quad (4)$$

The main advantage of this method is that the acquired instances in $N(c)$ are good representatives of the negatives that will have to be decoded, i.e., they are drawn from the same distribution $U(c)$. This allows the learning algorithm to focus on the typical properties of the incorrect examples extracted by the pool of KEs.

A drawback of this method is that instances in $N(c)$ are not guaranteed to be true negatives. It follows that the final training set may be noisy. Two main strategies can be applied to mitigate this problem: (1) Use a learning algorithm which is robust to noise in the training data; and (2) Adopt techniques to automatically reduce or eliminate noise. We here adopt the first solution, and leave the second as a possible avenue for future work, as described in Section 6. In Section 4 we demonstrate the amount of noise in our training data, and show that its impact is not detrimental for the overall performance of the system.

3 A Use Case: Entity Extraction

Entity extraction is a fundamental task in NLP (Cimiano and Staab, 2004; McCarthy and Lehnert, 2005) and web search (Chaudhuri et al., 2009; Hu et al., 2009; Tan and Peng, 2006), responsible for extracting instances of semantic classes (e.g., *'Brad Pitt'* and *'Tom Hanks'* are instances of the class *Actors*). In this section we apply our methods for automatically acquiring training data to the ES entity extraction system described in Pennacchiotti and Pantel (2009).[1]

The system relies on the following three **knowledge extractors**. $\boldsymbol{KE_{trs}}$: a 'trusted' database wrapper extractor acquiring entities from sources such as *Yahoo! Movies*, *Yahoo! Music* and *Yahoo! Sports*, for extracting respectively *Actors*, *Musicians* and *Athletes*. $\boldsymbol{KE_{pat}}$: an 'untrusted' pattern-based extractor reimplementing Paşca et al.'s (2006) state-of-the-art web-scale fact extractor. $\boldsymbol{KE_{dis}}$: an 'untrusted' distributional extractor implementing a variant of Pantel et al.'s (2009).

The system includes four **feature generators**, which compute a total of 402 features of various types extracted from the following sources: (1) a body of 600 million documents crawled from the Web at Yahoo! in 2008; (2) one year of web search queries issued to Yahoo! Search; (3) all HTML inner tables extracted from the above web crawl; (4) an official Wikipedia dump from February 2008, consisting of about 2 million articles.

The system adopts as a **ranker** a supervised Gradient Boosted Decision Tree regression model (GBDT) (Friedman, 2001). GBDT is generally considered robust to noisy training data, and hence is a good choice given the errors introduced by our automatic training set construction algorithms.

3.1 Training Data Acquisition

The positive and negative components of the training set for GBDT are built using the methods presented in Section 2, as follows:

Trusted positives (P_{trs} and P_{cls}): According to Eq. 2, we acquire a set of positive instances P_{cls} as a random sample of the instances extracted by both KE_{trs} and either: KE_{dis}, KE_{pat} or both of them. As a basic variant, we also experiment with the simpler definition in Eq. 1, i.e. we acquire a set of positive instances P_{trs} as a random sample of the instances extracted by the trusted extractor KE_{trs}, irrespective of KE_{dis} and KE_{pat}.

External positives (P_{cbc}): Any external repository of positive examples would serve here. In our spe-

[1] We here give a summary description of our implementation of that system. Refer to the original paper for more details.

cific implementation, we select a set of positive examples from the CBC repository (Pantel and Lin, 2002). CBC is a word clustering algorithm that groups instances appearing in similar textual contexts. By manually analyzing the cluster members in the repository created by CBC, it is easy to pick-up the cluster(s) representing a target class.

Same-class negatives (N_{cls}): We select a set of negative instances as a random sample of the instances extracted by only one extractor, which can be either of the two untrusted ones, KE_{dis} or KE_{pat}.

Near-class negatives (N_{oth}): We select a set of negative instances, as a random sample of the instances extracted by any of our three extractors for a class different than the one at hand. We also enforce the condition that instances in N_{oth} must not have been extracted for the class at hand.

Generic negatives (N_{cbc}): We automatically select as generic negatives a random sample of instances appearing in any CBC cluster, except those containing at least one member of the class at hand (i.e., containing at least one instance extracted by one of our KEs for the given class).

4 Experimental Evaluation

In this section, we report experiments comparing the ranking performance of our different methods for acquiring training data presented in Section 3, to three different baselines and a fully supervised upper-bound.

4.1 Experimental Setup

We evaluate over three semantic classes: *Actors* (movie, tv and stage actors); *Athletes* (professional and amateur); *Musicians* (singers, musicians, composers, bands, and orchestras), so to compare with (Pennacchiotti and Pantel, 2009). Ranking performance is tested over the test set described in the above paper, composed of 500 instances, randomly selected from the instances extracted by KE_{pat} and KE_{dis} for each of the classes[2].

We experiment with various instantiations of the ES system, each trained on a different training set obtained from our methods. The different system instantiations (i.e., different training sets) are reported in Table 1 (Columns 1-3). Each training set consists of 500 positive examples, and 500 negative examples.

As an **upper bound**, we use the ES system, where the training consists of 500 manually annotated instances (P_{man} and N_{man}), randomly selected from those extracted by the KEs. This allows us to directly check if our automatically acquired training sets can compete to the human upper-bound. We also compare to the following baselines.

Baseline 1: An unsupervised rule-based ES system, assigning the lowest score to instances extracted by only one KE, when the KE is untrusted; and assigning the highest score to any other instance.

Baseline 2: An unsupervised rule-based ES system, adopting as KEs the two untrusted extractors KE_{pat} and KE_{dis}, and a rule-based *Ranker* that assigns scores to instances according to the sum of their normalized confidence scores.

Baseline 3: An instantiation of our ES system, trained on P_{man} and N_{man}. The only difference with the upper-bound is that it uses only two features, namely the confidence score returned by KE_{dis} and KE_{pat}. This instantiation implements the system presented in (Mirkin et al., 2006).

For evaluation, we use *average precision* (AP), a standard information retrieval measure for evaluating ranking algorithms:

$$AP(L) = \frac{\sum_{i=1}^{|L|} P(i) \cdot corr(i)}{\sum_{i=1}^{|L|} corr(i)} \qquad (5)$$

where L is a ranked list produced by a system, $P(i)$ is the precision of L at rank i, and *corr(i)* is 1 if the instance at rank i is correct, and 0 otherwise.

In order to accurately compute statistical significance, we divide the test set in 10-folds, and compute the AP mean and variance obtained over the 10-folds. For each configuration, we perform the random sampling of the training set five times, rebuilding the model each time, to estimate the variance when varying the training sampling.

4.2 Experimental Results

Table 1 reports average precision (AP) results for different ES instantiations, separately on the three

[2] We do not test over instances extracted by KE_{trs}, as they do not go though the decoding phase

System	***Training Set***		***AP***			***MAP***
	Positives	**Negatives**	**Actors**	**Athletes**	**Musicians**	
Baseline1 (unsup.)	-	-	0.562	0.535	0.437	0.511
Baseline2 (unsup.)	-	-	0.676	0.664	0.576	0.639
Baseline3 (sup.)	P_{man}	N_{man}	0.715	0.697	0.576	0.664
Upper-bound (full-sup.)	P_{man}	N_{man}	0.860§	0.901§	0.786§	0.849§
S1.	P_{cls}	N_{oth}	0.751†	**0.880**§	0.642	0.758§
S2.	P_{cls}	N_{cbc}	0.734†	0.854§	0.644	0.744‡
S3.	P_{cls}	N_{cls}	**0.842**§	0.806§	**0.770**§	0.806§
S4.	P_{cls}	$N_{oth} + N_{cbc}$	0.756‡	0.853§	0.693‡	0.767§
S5.	P_{cls}	$N_{cls} + N_{oth}$	0.835§	0.807§	0.763§	0.802§
S6.	P_{cls}	$N_{cls} + N_{cbc}$	0.838§	0.822§	0.768§	**0.809**§
S7.	P_{cls}	$N_{cls} + N_{oth} + N_{cbc}$	0.838§	0.818§	0.764§	0.807§

Table 1: Average precision (AP) results of systems using different training sets, compared to two usupervised Baselines, a supervised Baseline, and a fully supervised upper-bound system. § indicates statistical significance at the 0.95 level wrt all Baselines. ‡ indicates statistical significance at the 0.95 level wrt Baseline1 and Baseline 2. † indicates statistical significance at the 0.95 level wrt Baseline1.

classes; and the mean average precision (MAP) computed across the classes. We report results using P_{cls} as positive training, and varying the negative training composition[3]. Systems S1-S3 use a single method to build the negatives. Systems S4-S6 combine two methods (250 examples from one method, 250 from the other), and S7 combines all three methods. Table 3 reports additional basic results when varying the positive training set composition, and fixing the best performing negative set (namely N_{cls}).

Table 1 shows that all systems outperform the baselines in MAP, with 0.95 statistical significance, but $S2$ which is not significant wrt *Baseline 3*. $S6$ is the best performing system, achieving 0.809 MAP, only 4% below the supervised upper-bound (statistically insignificant at the 0.95 level). These results indicate that our methods for automatically acquiring training data are highly effective and competitive with manually crafted training sets.

A class-by-class analysis reveals similar behavior for *Actors* and *Musicians*. For these two classes, the best negative set is N_{cls} (system $S3$), achieving alone the best AP (respectively 0.842 and 0.770 for *Actors* and *Musicians*, 2.1% and 1.6% points below the upper-bound). N_{oth} and N_{cbc} show a lower accuracy, more than 10% below N_{cls}. This suggest that the most promising strategy for automatically acquiring negative training data is to collect examples from the target class, as they guarantee to be drawn from the same distribution as the instances to be decoded. The use of near- and far-misses is still valuable (AP results are still better than the baselines), but less effective.

Negative set	***False Negatives***		
	Actors	**Athletes**	**Musicians**
N_{cls}	5%	45%	30%
N_{oth}	0%	10%	10%
N_{cbc}	0%	0%	15%

Table 2: Percentage of false negatives in different types of negative sets, across the three experimented classes (estimations over a random sample of 20 examples per class).

Results for *Athletes* give different evidence: the best performing negative set is N_{oth}, performing significantly better than N_{cls}. To investigate this contrasting result, we manually picked 20 examples from N_{cls}, N_{oth} and N_{cbc} for each class, and checked their degree of noise, i.e., how many false negatives they contain. Table 2 reports the results: these numbers indicate that the N_{cls} is very noisy for the *Athletes* class, while it is more clean for the other two classes. This suggests that the learning algorithm, while being robust enough to cope with the small noise in N_{cls} for *Actors* and *Musicians*, it starts to diverge when too many false negatives are presented for training, as it happens for *Athletes*.

False negatives in N_{cls} are correct instances extracted by one untrusted KE alone. The results in

[3] For space limitation we cannot report exhaustively all combinations.

Table 2 indicates that our untrusted KEs are more accurate in extracting instances for *Athletes* than for the other classes: accurate enough to make our training set too noisy, thus decreasing the performance of $S3$ wrt $S1$ and $S2$. This indicates that the effectiveness of N_{cls} decreases when the accuracy of the untrusted KEs is higher.

A good strategy to avoid the above problem is to pair N_{cls} with another negative set, either N_{cbc} or N_{oth}, as in $S5$ and $S6$, respectively. Then, when the above problem is presented, the learning algorithm can rely on the other negative set to compensate some for the noise. Indeed, when adding N_{cbc} to N_{cls} (system $S6$) the accuracy over *Athletes* improves, while the overall performance across all classes (MAP) is kept constant wrt the system using N_{cls} ($S3$).

It is interesting that in Table 2, N_{cbc} and N_{oth} also have a few false negatives. An intrinsic analysis reveals that these are either: (1) Incorrect instances of the other classes that are actual instances of the target class; (2) Correct instances of other classes that are also instances of the target class. Case (1) is caused by errors of KEs for the other classes (e.g., erroneously extracting *'Matthew Flynt'* as a *Musician*). Case (2) covers cases in which instances are ambiguous across classes, for example *'Kerry Taylor'* is both an *Actor* and a *Musician*. This observation is still surprising, since Eq. 3 explicitly removes from N_{cbc} and N_{oth} any correct instance of the target class extracted by the KEs. The presence of false negatives is then due to the low coverage of the KEs for the target class, e.g. the KEs were not able to extract *'Matthew Flynt'* and *'Kerry Taylor'* as actors.

Correlations. We computed the Spearman correlation coefficient r among the rankings produced by the different system instantiations, to verify how complementary the information enclosed in the training sets are for building the learning model. Among the basic systems $S1 - S3$, the highest correlation is between $S1$ and $S2$ ($r = 0.66$ in average across all classes), which is expected, since they both apply the principle of acquiring negative examples from classes other than the target one. $S3$ exhibits lower correlation with both $S1$ and $S2$, respectively $r = 0.57$ and $r = 0.53$, suggesting that it is complementary to them. Also, the best system $S6$ has higher correlation with $S3$ ($r = 0.94$) than with $S2$ ($r = 0.62$), indicating that in the combination of N_{cls} and N_{cbc}, most of the model is built on N_{cls}.

System	***Training Set***		***AP***			***MAP***
	Pos.	**Neg.**	**Act.**	**Ath.**	**Mus.**	
S3.	P_{cls}	N_{cls}	0.842	0.806	0.770	0.806
S8.	P_{trs}	N_{cls}	0.556	0.779	0.557	0.631
S9.	P_{cbc}	N_{cls}	0.633	0.521	0.561	0.571

Table 3: Comparative average precision (AP) results for systems using different positive sets as training data.

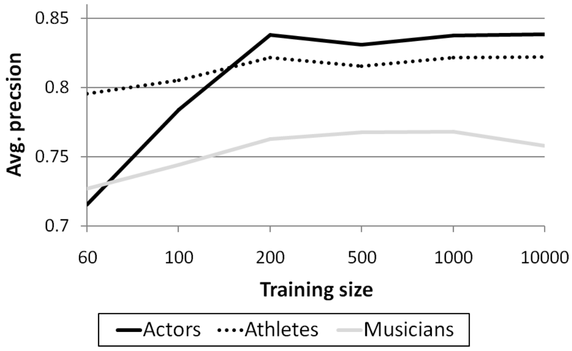

Figure 1: Average precision of system *S6* with different training sizes.

Varying the positive training. Table 3 reports results when fixing the negative set to the best performing N_{cls}, and exploring the use of other positive sets. As expected P_{cls} largely outperforms P_{trs}, confirming that removing the constraint in Eq. 2 and using the simpler Eq. 1 makes the training set unrepresentative of the unlabeled population. A similar observation stands for P_{cbc}. These results indicate that having a good trusted KE, or even an external resource of positives, is effective only when selecting from the training set examples that are also extracted by the untrusted KEs.

Varying the training size. In Figure 1 we report an analysis of the AP achieved by the best performing System ($S6$), when varying the training size, i.e., changing the cardinality of P_{cls} and $N_{cls} + N_{cbc}$. The results show that a relatively small-sized training set offers good performance, the plateau being reached already with 500 training examples. This is an encouraging result, showing that our methods can potentially be applied also in cases where few examples are available, e.g., for rare or not well-represented classes.

5 Related Work

Most relevant are efforts in *semi-supervised learning*. Semi-supervised systems use both labeled and unlabeled data to train a machine learning system. Most common techniques are based on co-training and self-training. *Co-training* uses a small set of labeled examples to train two classifiers at the same time. The classifiers use independent views (i.e. 'conditionally independent' feature sets) to represent the labeled examples. After the learning phase, the most confident predictions of each classifier on the unlabeled data are used to increase the labeled set of the other. These two phases are repeated until a stop condition is met. Co-training has been successfully applied to various applications, such as statistical parsing (Sarkar, 2001) and web pages classification (Yarowsky, 1998). *Self-training* techniques (or bootsrapping) (Yarowsky, 1995) start with a small set of labeled data, and iteratively classify unlabeled data, selecting the most confident predictions as additional training. Self-training has been applied in many NLP tasks, such as word sense disambiguation (Yarowsky, 1995) and relation extraction (Hearst, 1992). Unlike typical semi-supervised approaches, our approach reduces the needed amount of labeled data not by acting on the learning algorithm itself (any algorithm can be used in our approach), but on the method to acquire the labeled training data.

Our work also relates to the automatic acquisition of labeled negative training data. Yangarber et al. (2002) propose a pattern-based bootstrapping approach for harvesting generalized names (e.g., diseases, locations), where labeled negative examples for a given class are taken from positive seed examples of 'competing' classes (e.g. examples of diseases are used as negatives for locations). The approach is semi-supervised, in that it requires some manually annotated seeds. The study shows that using competing categories improves the accuracy of the system, by avoiding *sematic drift*, which is a common cause of divergence in boostrapping approaches. Similar approaches are used among others in (Thelen and Riloff, 2002) for learning semantic lexicons, in (Collins and Singer, 1999) for named-entity recognition, and in (Fagni and Sebastiani, 2007) for hierarchical text categorization. Some of our methods rely on the same intuition described above, i.e., using instances of other classes as negative training examples. Yet, the ES framework allows us to add further restrictions to improve the quality of the data.

6 Conclusion

We presented simple and general techniques for automatically acquiring training data, and then tested them in the context of the Ensemble Semantics framework. Experimental results show that our methods can compete with supervised systems using manually crafted training data. It is our hope that these simple and easy-to-implement methods can alleviate some of the cost of building machine learning architectures for supporting open-domain information extraction, where the potentially very large number of classes to be extracted makes infeasible the use of manually labeled training data.

There are many avenues for future work. Although our reliance on high-quality knowledge sources is not an issue for many head classes, it poses a challenge for tail classes such as 'wine connoisseurs', where finding alternative sources of high precision samples is important. We also plan to explore techniques to automatically identify and eliminate mislabeled examples in the training data as in (Rebbapragada and Brodley, 2007), and relax the boolean assumption of trusted/untrusted extractors into a graded one. Another important issue regards the discovery of 'near-classes' for collecting near-classes negatives: we plan to automate this step by adapting existing techniques as in (McIntosh, 2010). Finally, we plan to experiment on a larger set of classes, to show the generalizability of the approach.

Our current work focuses on leveraging auto-learning to create an extensive taxonomy of classes, which will constitute the foundation of a very large knowledge-base for supporting web search.

References

Avrim L. Blum and Pat Langley. 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. 1989. Proceedings of ltc-07. *Journal of ACM*, 36:929–965.

Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*, pages 875–883.

Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. 2009. Exploiting web search to generate synonyms for entities. In *Proceedings of WWW-09*, pages 151–160.

Philipp Cimiano and Steffen Staab. 2004. Learning by googling. *SIGKDD Explorations*, 6(2):24–34.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of WVLC/EMNLP-99*, pages 100–110.

Tiziano Fagni and Fabrizio Sebastiani. 2007. On the selection of negative examples for hierarchical text categorization. In *Proceedings of LTC-07*, pages 24–28.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*, pages 539–545.

Jian Hu, Gang Wang, Fred Lochovsky, Jian tao Sun, and Zheng Chen. 2009. Understanding user's query intent with Wikipedia. In *Proceedings of WWW-09*, pages 471–480.

N. Japkowicz and S. Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5).

M. Kubat and S. Matwin. 1997. Addressing the curse of inbalanced data sets: One-side sampleing. In *Proceedings of the ICML-1997*, pages 179–186. Morgan Kaufmann.

Joseph F. McCarthy and Wendy G Lehnert. 2005. Using decision trees for coreference resolution. In *Proceedings of IJCAI-1995*, pages 1050–1055.

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365, Massachusetts, USA. Association for Computational Linguistics.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of ACL/COLING-06*, pages 579–586.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of AAAI-06*, pages 1400–1405.

Marius Paşca. 2007. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*, pages 683–690, New York, NY, USA.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of KDD-02*, pages 613–619.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP-09*.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 238–247, Singapore. Association for Computational Linguistics.

Umaa Rebbapragada and Carla E. Brodley. 2007. Class noise mitigation through instance weighting. In *Proceedings of the 18th European Conference on Machine Learning*.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *NAACL-2001*.

Bin Tan and Fuchun Peng. 2006. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of WWW-06*, pages 1400–1405.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 214–221, Philadelphia, PA, USA. Association for Computational Linguistics.

Richard C. Wang and William W. Cohen. 2008. Iterative set expansion of named entities using the web. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, Washington, DC, USA. IEEE Computer Society.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised learning of generalized names. In *COLING-2002*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-1996*, pages 189–196.

David Yarowsky. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory*, pages 92–100.