# PORE: Positive-Only Relation Extraction from Wikipedia Text[*]

Gang Wang, Yong Yu and Haiping Zhu

Apex Data & Knowledge Management Lab,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
{gavinwang,yyu,zhu}@apex.sjtu.edu.cn

**Abstract.** Extracting semantic relations is of great importance for the creation of the Semantic Web content. It is of great benefit to semi-automatically extract relations from the free text of Wikipedia using the structured content readily available in it. Pattern matching methods that employ information redundancy cannot work well since there is not much redundancy information in Wikipedia, compared to the Web. Multi-class classification methods are not reasonable since no classification of relation types is available in Wikipedia. In this paper, we propose *PORE* (*Positive-Only Relation Extraction*), for relation extraction from Wikipedia text. The core algorithm *B-POL* extends a state-of-the-art *positive-only learning* algorithm using bootstrapping, strong negative identification, and transductive inference to work with fewer positive training examples. We conducted experiments on several relations with different amount of training data. The experimental results show that *B-POL* can work effectively given only a small amount of positive training examples and it significantly outperforms the original positive learning approaches and a multi-class SVM. Furthermore, although *PORE* is applied in the context of Wikipedia, the core algorithm *B-POL* is a general approach for Ontology Population and can be adapted to other domains.

**Keywords:** Relation Extraction, Ontology Population, Positive-Only Learning

## 1 Introduction

The Semantic Web builds on not only ontologies but also the contents conforming to the ontologies. According to a recent study [1], although Semantic Web data is growing steadily on the Web, the space of instances is sparsely populated (most classes (>97%) have no instances and the majority of properties (>70%) have never been used to assert data). Consequently, Ontology Population and Annotation are of great importance for the realization of the Semantic Web.

It is of great benefit to extract semantic content from Wikipedia, the largest free online encyclopedia (http://www.wikipedia.org). Völkel et al. [12] provided an extension to be integrated into Wikipedia to allow the creation of an open semantic knowl-

[*] This work is funded by IBM China Research Lab.

edge base. Auer and Lehmann [13] recently argued that means for creating semantically enriched structured content are already available and used by Wikipedia authors. They presented a pattern-matching approach to extract the structured content and proposed strategies that require only minor modifications of the wiki systems for improving the quality of the creation of structured content. In this paper, we plan to go a step further. We propose an approach that exploits the structured content readily available in Wikipedia to semi-automatically extract semantic relations between Wikipedia entities from the free text. The relations are already defined in the structured tables, along with a set of relation instances. This is an Ontology Population task, where only a relatively small amount of relation instances are available for learning while no negative examples are provided.

A great amount of research work has been conducted to extract relations using a small amount of seed instances. DIPRE [3] paradigm based work Snowball [4], Espresso [5], and [6], etc. employed bootstrapping based pattern matching approaches. The approaches exploited *information redundancy* of the Web — instances to be extracted will tend to appear in uniform contexts repeatedly. However, compared to the Web, information redundancy cannot be guaranteed in Wikipedia.

Work conducted in [19] [20] performed multi-class relation classification [21] based on a hierarchical classification of relation types. However, relations to be extracted from Wikipedia are more fine-grained and diverse so that no such relation type classification is available in Wikipedia. Consequently, it is not reasonable to employ multi-class classification.

In this paper, we propose *PORE* (*Positive-Only Relation Extraction*), a new approach to extracting relation instances from Wikipedia text. The core algorithm *B-POL* builds on top of a state-of-the-art *positive-only learning* (*POL*) approach [14] [15] that initially identifies strong negative examples from unlabeled data and then iteratively classifies more negative data until convergence. *B-POL* makes several extensions to *POL* to work with fewer positive examples without sacrificing too much precision. Specifically, a conservative strategy is made to generate strong initial negative examples, resulting in high recall at the first step. The newly generated positive data identified by *POL* are added for training and the underlying *POL* approach is invoked again to generate more positive data. The method iterates until no positive data can be generated anymore. It exploits unlabeled data for learning and is transformed to a transductive [22] learning method that is believed to work better with sparse training data. Furthermore, it is built on top of a state-of-the-art statistical learning algorithm SVM. These settings enable the effective learning with fewer positive examples. To the best of our knowledge, no work has been done on using positive-only learning (classification) algorithms for relation extraction.

We conducted experiments on several relations, each of which has different amount of training instances. We evaluated the results against a manually constructed *gold standard* and it showed that the core algorithm *B-POL* outperforms a simple transductive version of *POL* and a transductive *POL* with a conservative strategy. *B-POL* also significantly outperforms a multi-class SVM approach. Last but not least, although *PORE* is applied in the context of Wikipedia, the core algorithm *B-POL* is a general approach for Ontology Population and can be adapted to other domains.

The rest of the paper is organized as follows. Section 2 compares our work with other ongoing relevant research work. In Section 3, we elaborate on the core algorithm, *B-POL*. We give in Section 4 the description of Wikipedia and the features, as well as the filtering process. Section 5 describes the experiments and evaluation. Finally, we conclude this paper and present future work in Section 6.

## 2 Related Work

DIPRE [3] based methods [4] [5] [6] exploited information redundancy on the Web and the pattern/relation duality by using pattern matching combined with bootstrapping. Exploring the Web for redundancy information is reasonable. However, such systems need to estimate the confidence of patterns and instances, which is a rather difficult task. Other methods exploiting information redundancy can be found in [7] [8]. These systems generally face the problem that many parameters need to be specified for each relation.

LEILA [18] automatically generated negative examples using information about the cardinality of relations. Work conducted in [19] [20] employed semi-supervised learning algorithms and achieved good performance using only a small amount of labeled examples. They performed multi-class classification in which all the relation types are already defined [21]. Mori et al. [9] described an approach for extracting relations in social networks. Work by Wang et al. [21] was conducted on the ACE corpus using various features. Schutz and Buitelaar [27] described *RelExt* for extracting relations in the football domain. Tang et al. [10] proposed Tree-CRF for semantic annotation on semi-structured data. Ramakrishnan et al. [2] described a schema-driven approach to relation extraction from biomedical text.

The *Semantic Wikipedia* project described in [12] provided an extension to be integrated into Wikipedia to allow the creation of an open semantic knowledge base. A recent study [13] directly extracted structured tables of relations from Wikipedia using pattern matching. YAGO [31] built an ontology by extracting relations from Wikipedia categories. It mainly employed heuristic rules and WordNet during the extraction and presented results of high quality. However, the approach is somehow limited to the extraction of certain types of relations due to the fact that it did not explore the free text which is the main source of relations. Ruiz-Casado et al. [25] described an extraction pattern-based method for extracting *is-a* and *part-whole* relations from Wikipedia text to enrich WordNet. Wang et al. [24] exploited various features in Wikipedia to enhance the extraction of relations from Wikipedia text. However, the method requires manual tuning of the similarity thresholds for each pattern, which is tedious and impractical for large scale applications. In this paper, we employ feature-based SVM classification [26], which is believed to be more robust, to extract mainly non-taxonomic relations from Wikipedia.

## 3 B-POL

In this section, we present the core algorithm, *B-POL*. It builds on top of two similar state-of-the-art *positive-only learning* approaches PEBL [14] and Roc-SVM [15] that initially identify strong negative examples from unlabeled data and then iteratively classify more negative data until no such data can be found.

Prior to the illustration of the learning framework, we first formulate the relation extraction problem as a positive-only binary classification task.

Given a collection $C$ of co-occurrence contexts of entity pairs, a given relation type $R$ as well as a set of entity pairs as training data (the corresponding co-occurrence contexts in $C$ are denoted as $P$, the positive set), the task is to assign the relation type $R$ to occurrences (in the unlabeled set $U = C - P$) that indicate the relation. (Each co-occurrence context is represented as a vector of relevant features which are explained in Sec. 4)

The original positive-only classification method proposed in [14] and [15] is an inductive learning algorithm [22] because they output a final classifier that can make predictions on unseen data. Since it is believed that transductive inference is generally suited to the problems with a small amount of training data [22], we transformed the original method into a transductive one. We call the adaptation of the positive-only learning method as *T-POL* (*Transductive Positive-Only Learning*), which is shown in Fig. 1.

Algorithm: *T-POL* ($P$, $U$)
Input: positive set $P$, unlabeled set $U$
Output: a set $P_u$ of examples finally classified as positive

1. Use a weak classifier $\Psi$ to classify using $P$ and $U$. The data in $U$ classified as positive is $P_0$, the strong negatives $N_0 \leftarrow U - P_0$
2. Set $N \leftarrow \Phi$, $i \leftarrow 0$
3. Do loop
   3.1 $N \leftarrow N \cup N_i$
   3.2 Use $\nu$-SVM to classify $P_i$ with positive set $P$ and negative set $N$
      3.2.1 $N_{i+1} \leftarrow$ examples from $P_i$ classified as negative
      3.2.2 $P_{i+1} \leftarrow$ examples from $P_i$ classified as positive
   3.3 $i \leftarrow i + 1$
   3.4 Repeat until $N_i = \Phi$
4. $P_u \leftarrow P_i$ , return $P_u$

**Fig. 1.** Transductive Positive-Only Learning method (*T-POL*)

In step 1 of *T-POL* algorithm, a weak classifier $\Psi$ is employed to draw an initial approximation of "strong negatives", which are the negative data located far from the boundary of the positive class in the universal feature space. Rocchio [16] and OSVM (One-Class SVM) [28] were employed as the weak classifier $\Psi$ in [15] and [14], respectively. In step 3.2 of *T-POL*, $\nu$-SVM [23] is employed to maximize the margin using the positive data and the current version of negatives. $\nu$-SVM is a version of SVM with a soft margin and is necessary for *T-POL* to cope with noises in the training data [14]. The rate of noise in training data is controlled by the parameter $\nu$, which can generally be set to a low value (e.g. 0.01). $\nu$-SVM maximizes the margin at each iteration and thus progressively improves the approximation of negative data. Consequently, the class boundary eventually converges to the true boundary of the positive class in feature space [15].

However, in step 1, the weak classifier $\Psi$ in [15] and [14] tends to generate too many false negatives from $U$, which results in low recall in later iterations. As pointed out in [14], classifier $\Psi$ should generate pure negatives $N_0$ excluding false negatives by sacrificing precision in $P_0$. The precision of step 1 does not affect the accuracy of the final boundary as far as it approximates a certain amount of negative data because the final boundary will be determined by step 2-4. Motivated by this, we only select the "strongest" negatives identified by $\Psi$. The modified classifier $\Psi$ based on Rocchio is named *Roc-SN*, which is shown in Fig. 2.

Algorithm: *Roc-SN* (*P*, *U*, *c*)
Input: positive set *P*, unlabeled set *U*, the percentage *c* of the "strongest" negatives out of all negatives identified by Rocchio.
Output: a set $N_0$ of "strongest" negatives
--- Each instance is represented as $i$, with corresponding vector $\vec{i}$

1. Construct two prototype vectors:
   1.1 $\vec{c}^{+} \leftarrow \alpha \frac{1}{|P|} \sum_{i \in P} \frac{\vec{i}}{\|\vec{i}\|} - \beta \frac{1}{|U|} \sum_{i \in U} \frac{\vec{i}}{\|\vec{i}\|}$
   1.2 $\vec{c}^{-} \leftarrow \alpha \frac{1}{|U|} \sum_{i \in U} \frac{\vec{i}}{\|\vec{i}\|} - \beta \frac{1}{|P|} \sum_{i \in P} \frac{\vec{i}}{\|\vec{i}\|}$
2. Set $N_0 \leftarrow \Phi$
3. For each instance $i$ in $U$ do loop
   3.1 $s_i \leftarrow sim\left(\vec{c}^{-}, \vec{i}\right) - sim\left(\vec{c}^{+}, \vec{i}\right)$
   3.2 If $s_i > 0$ then $N_0 \leftarrow N_0 \cup i$
4. $N_0 \leftarrow$ top $\lfloor c \times |N_0| \rfloor$ instances with largest $s_i$, return $N_0$

**Fig. 2.** Modified version of Rocchio for identifying the "strongest" negatives (*Roc-SN*)

In Rocchio classification, the classifier is built by constructing positive and negative prototype vectors (the unlabeled data are treated as negatives). If the similarity (*sim*, cosine similarity) between the test instance *i* and the negative prototype vector is larger than that between *i* and the positive one, *i* is added to negative set. The parameters $\alpha$ and $\beta$ adjust the relative impact of positive and negative instances and are set to 16 and 4, respectively in text classification tasks [15]. In *Roc-SN*, $s_i$ is used to measure the "strength" of the negative instance *i*. Parameter *c* is used to determine the percentage of instances that are selected as the "strongest" negatives out of the entire set of negatives identified by Rocchio. In this way, the top $\lfloor c \times |N_0| \rfloor$ negatives with the largest "strength" are finally retained in *Roc-SN*. The smaller *c* is, the purer the generated "strongest" negatives are. This means a smaller *c* could generally bring higher recall while a larger *c* would give higher precision as it identifies more negatives. It is obvious that it degenerates to the original Rocchio classifier when $c = 1$.

However, when the positive examples are too few, *T-POL* would end up fitting tightly around the few positive training examples, resulting in low recall [14]. Having observed that precision is not directly influenced when positive examples are under-sampled, we extend *T-POL* by adding the positive data ($P_u$) newly generated by *T-POL* to the set of training examples and invoking *T-POL* again to generate more positive data. The algorithm iterates until no positive data can be returned from *T-*

*POL*. This bootstrapping version of *T-POL* gives the core algorithm, *B-POL*, which is illustrated in Fig. 3.

Algorithm: *B-POL* (*P*, *U*)
Input: positive set *P*, unlabeled set *U*
Output: a set $P_u$ of examples classified as positive

1. Set $P_u \leftarrow \Phi$, $i \leftarrow 0$
2. Do loop
   2.1 $i \leftarrow i + 1$
   2.2 Set $P_u^{(i)} \leftarrow$ positive examples returned from $T\text{-}POL(P \cup P_u, U)$
   2.3 $P_u \leftarrow P_u \cup P_u^{(i)}$ , $U \leftarrow U - P_u^{(i)}$
   2.4 Repeat until $P_u^{(i)} = \Phi$
3. Return $P_u$

**Fig. 3.** Bootstrapping POL (*B-POL*)

This kind of bootstrapping is commonly referred to as self-training, which has been reported to perform well especially in natural language processing tasks [30]. In *B-POL*, the classifier uses its own predictions to re-train itself. It is such reinforcement that contributes to the high recall when fewer positive examples are provided.

# 4 PORE

*PORE* works as follows: 1) extracting entity features from semi-structured data of Wikipedia; 2) extracting entity-pair co-occurrence context from Wikipedia text; 3) for each relation, filtering out irrelevant instances using the positive training data extracted from the structured content of Wikipedia; 4) conducting relation classification on the filtered set of instances using *B-POL*. The positive instances output by *B-POL* are manually examined and the true positives are finally stored as RDF triples.

## 4.1 Wikipedia

Wikipedia is a hypertext document collection with a rich link structure. Generally in each page of Wikipedia, the first sentence serves as the definition of an entity (entry). The *bold italic* phrase in the definition is a self-reference to the current entry. Each article in Wikipedia is assigned at least one category. In some articles, an infobox containing a picture gives a general description of an entity. In each infobox within an article, there are a set of properties defined to describe the entity. Each property generally demonstrates a relation between two entities. The entity described by the current article can be viewed as the subject of the relations. The objects are connected by relation predicates and are mainly internal links that point to other entities in Wikipedia or just literal text or, in some cases, external links pointing to web pages outside Wikipedia. Fig. 4 gives a snapshot of the article "*Annie Hall*", which demonstrates the (semi-) structured contents associated with a Wikipedia entry.

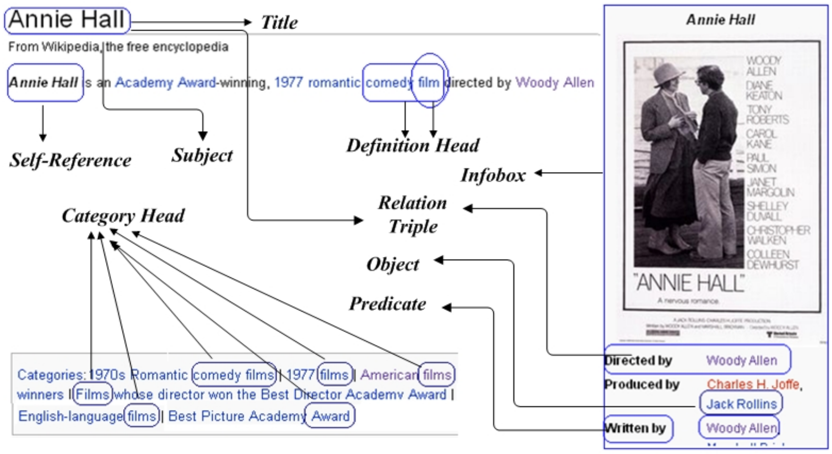


**Fig. 4.** (Semi-)Structured Contents in Wikipedia (from entry *Annie Hall*).

## 4.2 Feature Engineering

Feature based relation extraction using SVM is a popular approach and gives the current best reported results on ACE corpus in [26]. We separate *entity features* which describe Wikipedia entities from *context features* which describe co-occurrence contexts of pairs of Wikipedia entities.

***Entity Feature Extraction.*** As shown in Fig. 4, a Wikipedia entity (entry) is described by definition, categories as well as predicates in the infobox. Wang et al. [24] argued that the Wikipedia entity features are more powerful than traditional Named Entity Recognition (NER) since they give more fine-grained descriptions for an entity.

For *definition features*, we heuristically extract the head word of the first base noun phrase (BNP) following a *be*-verb (i.e. *is, was, are, were*, etc.). For example, in the sentence "*Annie Hall is an Academy Award-wining, 1997 romantic comedy_film directed by Woody Allen.*", the word "*film*" and the augmented word "*comedy_film*" are extracted as entity features for the Wikipedia entity "*Annie Hall*".

For *category feature*s, since the name of each category is a noun phrase, heuristically, the head word of the first base noun phrase in the category phrase is extracted. Take the entry "*Annie Hall*" for example, "*film*" and the augmented version "*comedy_film*" are extracted from category "*Romanic comedy films*".

For *infobox features*, names of the predicates, with each white space character replaced by an underscore (e.g. "*produced_by*", "*written_by*", etc.) are kept.

***Context Feature Extraction.*** Context features are derived from the co-occurrence of entity pairs in a sentence. As in the sentence "*In the film "Heavenly Creatures", directed by Peter Jackson, Juliet Hulme had TB, and her fear of being sent …*", there are three hyperlinked entities (as indicated by the underscore). For each pair of entities, e.g. (*Heavenly Creatures , Peter Jackson)*, tokens to the left of *Heavenly Creatures*, those to the right of *Peter Jackson*, and those in between the two entities are extracted and encoded as the context features. For the details of how to encode the context features using the tokens, one may refer to the technical report [29] which provides a formal definition of the features.

### 4.3 Data Filtering

The number of the entity pairs can be very large, and thus it is inefficient if they are directly classified. Furthermore, because of the highly skewed data distribution, the recall of the SVMs would decrease. In Snowball [4], named entity types of a relation are used to filter data. In the same way, we use the entity features for filtering.

We first define a feature selection method. We denote the complete set of data as $C$ and the positive set in $C$ as $P$. To define a score of a feature $f$, we further denote the set of data from $P$ containing $f$ as $P_f$ and the set of data from $C$ containing $f$ as $C_f$. The feature scoring function is shown in equation (1).

$$score(f) = |P_f| \times \log\left(|C| / |C_f|\right). \tag{1}$$

It can be observed that features of an entity are usually diverse, expressing different aspects of the entity. Nevertheless, it is reasonable to assume that entities in a given relation at a given argument position (subject or object) share a certain degree of commonality [24]. We use equation (1) to score features of entities at each argument position (subject or object) and select top $k$ features with the highest scores. The value of $k$ is set according to the following heuristics:

- $k = \lfloor 10\% * \#\text{entity features} \rfloor$, (if $k = 0$, then $k = 1$; if $k > 15$, then $k = 15$).

The selected features are called *Salient Entity Features*. For convenience, the salient features of entities at subject (object) position are called *Salient Subject* (*Object*) *Features*. The set of entity pairs from which features of the left-hand-side entity intersect with the *Salient Subject Features* and meanwhile features of the right-hand-side entity intersect with the *Salient Object Features* are kept. We denote the set of entity pairs finally kept as $C'$, and then the unlabeled set $U = C' - P$. Finally, we apply *B-POL* to classify $U$ using $P$ (see Sec. 3).

As in the previous example, "*In the film "Heavenly Creatures", directed by Peter Jackson, Juliet Hulme had TB, and her fear of being sent* ...", although there are several pairs of entities, pairs such as <Peter Jackson , Juliet Hulme>, <Heavenly Creatures , Juliet Hulme>, etc. will be filtered out when we are extracting *film-director* relation. This is because the *Salient Subject Features* and the *Salient Object Features* constructed using the positive training data are <*film, drama_film, movie*, ...> and <*director, film_director*, ...>, respectively, which do not have intersection with those of the filtered-out pairs.

## 5 Evaluation

For the experiments, we used the data from the *Wikipedia XML* corpus [17]. Our work is concerned with extracting relations between Wikipedia entities and thereby only the internal links are considered. In the current experimentation, the definitions of relations as well as the corresponding training instances come from the infoboxes of Wikipedia. Nevertheless, one can still define other relations and provide corresponding training instances to make *PORE* work. Here we focus on extracting relations from free text, so the highly structured pages with titles like "List of" or "Lists of" and the disambiguation pages are not considered. We finally obtained 644,508 pages.

In the experiments, all NLP tasks are performed using the OpenNLP toolkit (http://opennlp.sourceforge.net/). Stemming is performed by Snowball stemmer shipped with Lucene (http://lucene.apache.org). In the context feature extraction, we only keep links whose anchor text represents a proper noun.

We focus on evaluating the performance of the core part, *B-POL*. Two methods are selected as baselines. One is the simple transductive version of the original positive-only learning method using Rocchio[1], namely *T-POL'*. The other is *T-POL* with the modified Rocchio, namely *Roc-SN*.

The experiments are conducted over a subset of 10,000 pages randomly selected from the *Wikipedia XML* corpus. There are about 130,000 pairs of entities in the subset. In order to evaluate the performance using precision, recall and F1, we need to construct a *gold standard* set from the selected subset of pages for each relation. However, it is impractical to manually label the 130,000 pairs. Neither can we randomly sample a smaller subset since the distribution of the target relations is highly skewed. We also use the Wikipedia entity features to pre-filter the irrelevant pairs like what we do in Sec. 4.3. However, we do not directly take the original method in Sec. 4.3 since it is part of our approach to be evaluated. In contrast, we use the entire set of entity features. The construction of the *gold standard* is illustrated as follows.

1. Use Lucene to build an inverted index of the entity pairs using the entity features.
2. For each relation, we obtain a set of instances from the corresponding Wikipedia infoboxes. Then we find out the instance occurrences in the inverted index. Taking the occurrences as the positive set $P$ and the entire entity pairs as the unlabeled set $U$, we use equation (1) to calculate scores of the subject (object) features.
3. Use Lucene to build a BooleanQuery *subject_query (object_query)* by selecting all the subject (object) entity features as query terms and taking the corresponding feature scores (calculated using equation (1)) as query weights. A final BooleanQuery in the form of "*subject_query* AND *object_query*" is submitted to Lucene.
4. From the ranked list of entity pairs, we retain the top 1000 pairs only.
5. The 1000 pairs are manually examined by three human subjects. The correct entity pairs that achieve agreements, along with their co-occurrence context, are added to the *gold standard*.

For the algorithms in the experiments, we use LibSVM [11] which supports $\nu$-SVM (Sec. 3) to implement the *POL* methods. In terms of the specific SVM model, we choose RBF (Radial Basis Function) kernel. According to [11], it can handle non-linear relations between class label and attributes, and it subsumes linear kernel. In the experiments, $\nu$ of $\nu$-SVM is set to a theoretically motivated fixed parameter, 0.01 (Sec. 3). We use the default parameters provided in LibSVM for other parameter settings.

Prior to the demonstration of the results, we introduce the following denotations.

- P: the set of training data (entity pair occurrences).

[1] As mentioned in Sec. 3, transductive inference is believed to perform better than the inductive counterpart when handling small amount of training data. As a result, we do not take the original inductive one as a baseline.

- U: the set of the unlabeled instances after filtering (test data).
- GS: the set of instances in the *gold standard*.

In the infoboxes of the *Wikipedia XML* collection, there are currently 9,197 relations and 953,550 relation instances. Considering both the time and space limitations, currently we only select several relations for demonstration. The selection criteria are as follows: 1) there are a sufficient number of ground truth instances in the remaining data after filtering, making it possible to show the performance with different amount of training data; 2) the relations are somehow typical, so they can reflect different aspects of problems which need to be addressed; 3) the relations are from different domains.

Table 1 gives the information about the four relations that are tested in our experiments. The “Source” means the infobox from which the relation and its instances are extracted. #(GS ∩ U) indicates the performance of the data filtering using the *Salient Entity Features*. It can be seen that recall (calculated by #(GS ∩ U) / #GS) is relatively high. Precision at this stage does not matter much since the unlabeled data will be tested by *B-POL*.

**Table 1.** Information about the four relations.

| Relation | Source | #GS | #U | #(GS ∩ U) |
|---|---|---|---|---|
| album-artist | album_infobox#artist | 274 | 392 | 260 |
| film-director | infobox_movie#director | 121 | 286 | 115 |
| university-city | infobox_university#city | 74 | 208 | 71 |
| band-member | infobox_band#current_members | 117 | 477 | 103 |

Fig. 5 demonstrates the performance of *T-POL* and *B-POL* with different number of training data and different settings for parameter *c* (Sec. 3). F1-scores are plotted with different values of parameter *c* (it is used in each invocation of *Roc-SN*). It is obvious that the results of *T-POL* at $c = 1$ are also the results of *T-POL’*. Each value of F1-score is averaged over 20 trials. Table 2 gives the results of *B-POL* and *T-POL*. At each invocation of *Roc-SN*, *c* is set to a random value that ranges from 0.1 to 1.0. The results in Table 2 are averaged over 50 trials to achieve high reliability.

***B-POL* vs *T-POL* and *T-POL’***. From Fig. 5, especially the results of *album-artist*, *film-director* and *band-member* relations, *B-POL* consistently outperforms *T-POL* at nearly all settings of parameter *c*. The averaged F1-scores in Table 2 also demonstrate the significant improvement of *B-POL* over *T-POL* and *T-POL’*. *B-POL* significantly increases recall by sacrificing not too much precision.

For the *university-city* relation, the gap is smaller. F1-scores of *T-POL’* and *T-POL* even surpass that of *B-POL* when #P=40. This is largely due to the reason that when the number of the positives in the unlabeled data is small, the bootstrapping strategy of *B-POL* would not benefit from the improvements in recall but just lowering precision. As described in Table 1, the size of the *gold standard* of *university-city* is small. When #P=40, the number of positives in the unlabeled set is smaller than #P. In this case, the original *POL* methods work better. We also found that the co-occurrence contexts for *university-city* relation are quite general (e.g. “*<university>, <city>*”, “*<university> in <city>*”, “*<university>, at <city>*”). The bootstrapping strategy of *B-POL* brings more errors during further iterations, which results in much decrease in precision. However, the increase of recall brought by *B-POL* produces larger F1-scores when the amount of training data is much less (i.e. #P=10).

From Table 2, it can be observed that *B-POL* achieves significantly higher F1-scores than *T-POL* and *T-POL'* when the amount of the training data is less. This indicates that *B-POL* is more effective when dealing with fewer positive training data.

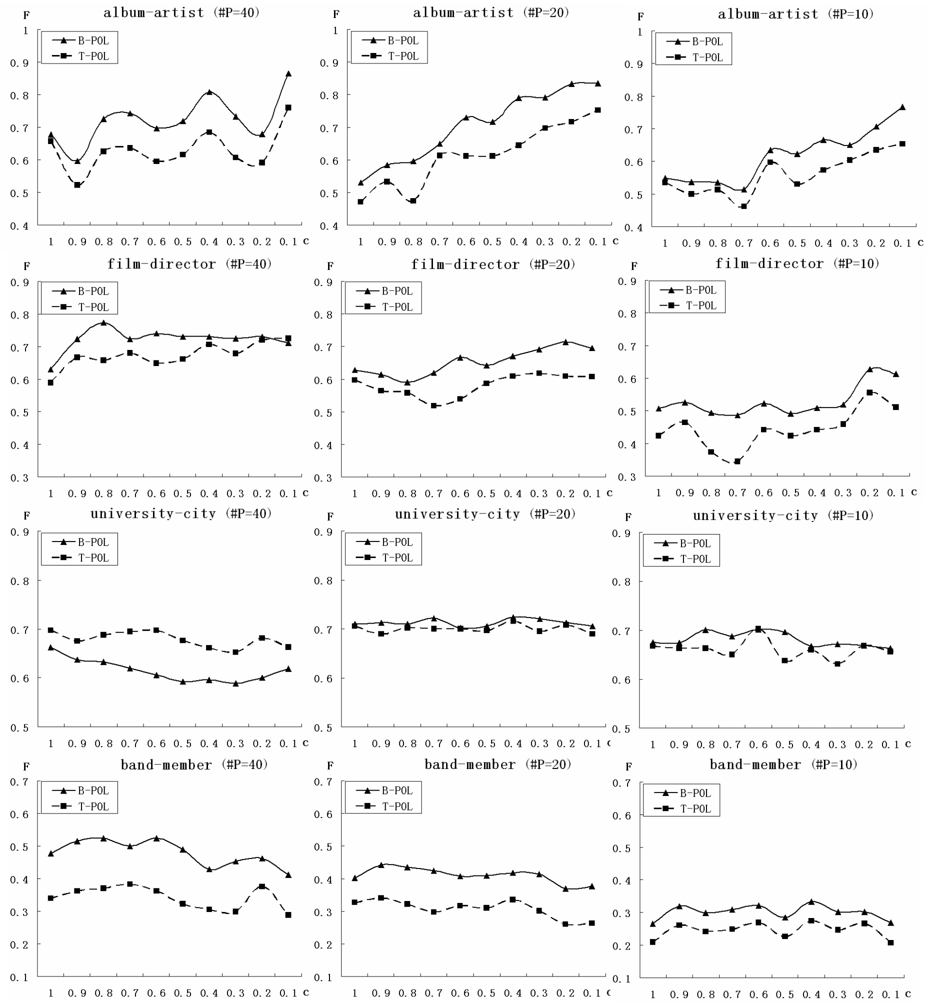


**Fig. 5.** F1-scores of *T-POL* and *B-POL* on the four relations with different settings.

***POL* vs *M-SVM*.** We also assess the performance of multi-class classification using LibSVM (*M-SVM*). We use the same training data and unlabeled data in *B-POL* for *M-SVM*. Our original setting for *M-SVM* is as follows. We treat each of the four relations as a class and add another the "others" class to indicate other relations or unrelated entity pairs. The examples for "others" class are sampled in the entire collection of entity pairs excluding the portion in the *gold standard* of the four relations. The sample size of the "others" class is equal to that of the four relations. However, this setting produces rather bad performance. Even when the same amount of training data is used for each class, the *album-artist* relation and "others" are always overwhelming and *M-SVM* just distributes the labels of the unlabeled data to the two classes. The other three relations obtain nearly zero F1-scores. Consequently, we actually conduct two-class classification by each time selecting only one of the four relations. The results are better than that of the original one. However, it can be seen in Table 2, the performance of *M-SVM* is still worse than *B-POL* and *T-POL*. It is even worse than *T-POL'* when the training data is not much under-sampled in most cases.

**Table 2.** The extraction performance (Prec./Rec./F1) of *B-POL* and the other 3 baselines.

| #P | method | album-artist | film-director | university-city | band-member |
|---|---|---|---|---|---|
| | | P/R/F1 | P/R/F1 | P/R/F1 | P/R/F1 |
| 40 | *T-POL'* | 96.7/36.5/47.8 | 82.8/50.6/60.6 | 65.4/74.4/**68.6** | 70.2/25.0/35.7 |
| | *T-POL* | 89.6/49.8/59.2 | 82.2/58.2/66.4 | 62.0/76.8/68.1 | 67.6/25.0/34.8 |
| | *B-POL* | 86.6/77.5/**79.9** | 69.4/81.2/**73.2** | 47.2/84.8/58.5 | 46.8/57.6/**47.1** |
| | *M-SVM* | 93.6/40.4/54.5 | 71.2/32.8/41.4 | 17.4/36.9/19.5 | 35.4/29.7/ 27.5 |
| 30 | *T-POL'* | 97.4/45.8/58.8 | 85.5/51.1/62.2 | 75.1/67.7/70.5 | 74.3/24.5/35.9 |
| | *T-POL* | 93.2/56.7/68.2 | 83.7/51.0/61.8 | 70.7/72.6/**70.6** | 67.6/22.0/32.4 |
| | *B-POL* | 90.6/70.2/**76.5** | 73.4/69.6/**68.6** | 62.7/79.0/68.5 | 58.5/46.6/**49.3** |
| | *M-SVM* | 93.4/46.2/58.0 | 72.1/37.9/44.8 | 20.9/33.7/21.9 | 36.1/32.5/30.0 |
| 20 | *T-POL'* | 97.1/34.6/48.0 | 84.6/37.7/49.9 | 80.3/63.6/70.5 | 77.7/21.7/33.5 |
| | *T-POL* | 93.5/52.8/63.7 | 81.3/47.0/56.5 | 79.8/64.0/70.2 | 72.3/21.0/31.5 |
| | *B-POL* | 90.0/69.2/**76.4** | 74.7/64.1/**66.6** | 75.3/70.1/**71.6** | 67.9/32.3/**41.9** |
| | *M-SVM* | 93.8/42.4/55.9 | 73.1/40.5/46.9 | 27.0/31.6/26.0 | 39.4/32.9/29.8 |
| 10 | *T-POL'* | 99.1/35.3/50.7 | 89.1/32.1/45.7 | 82.5/57.7/66.7 | 81.4/12.5/21.2 |
| | *T-POL* | 96.7/40.5/53.8 | 86.2/30.5/42.5 | 84.1/54.1/64.8 | 76.7/15.2/24.6 |
| | *B-POL* | 95.0/48.6/**61.3** | 83.2/41.3/**51.0** | 82.7/58.1/**67.5** | 74.0/19.9/**30.1** |
| | *M-SVM* | 93.4/46.3/58.9 | 78.3/31.4/42.7 | 32.1/28.1/29.1 | 40.6/32.8/26.4 |

Note that we actually feed additional "negative" information to *M-SVM* by providing the "others" class with the sampled data that are known to be absent from the *gold standard* of the four relations. However, in most cases, the performance of *M-SVM* is still poorer than the "*POL*" methods. On one hand, since it is believed that unlabeled data can significantly help learning [14] [15], it is intuitive for one to expect that the "*POL*" methods that employ the unlabeled data in learning outperform *M-SVM* that does not. On the other hand, the "*POL*" methods are transductive, which is believed to be better than the inductive one, *M-SVM*, when dealing with sparse training data [22].

**Impact of *c*.** Looking at Fig. 5, we can observe that *B-POL* and *T-POL* obtain significantly higher F1-scores on *album-artist* relation when parameter *c* is smaller. This is because lower *c* settings conservatively identify smaller portion of negatives that are strongest in *Roc-SN* (Sec. 3), which results in greatly improved recall. The results on *film-director* relation are similar but the changes of F1-scores are less significant along the different settings of *c*. The results on the other two relations, excluding *band-member* (#P=40), do not change much with different settings of *c*. In *band-member* (#P=40), the precision decreases too much when the smaller amount of negatives identified by *Roc-SN* cannot cover a sufficiently large region. From the investigations, we found that *album-artist* and *film-director* relations are described by strong co-occurrence contexts while those of the other two relations are somehow general. In the cases of strong contexts, the precision would not decrease much when strategies are made to increase recall. Nevertheless, for general contexts, to a certain degree, recall is already guaranteed by the contexts, so the decrease of precision dominates the F1-scores when lower *c* values are set.

**Efficiency.** As described in [15], the time complexity of the original *POL* is $O(|U|^2 * log|U|)$, assuming the number of iterations is $log|U|$ (here *U* represents the set of unlabeled instance). For *B-POL*, although the number of iterations invoking *T-POL* cannot be pre-determined, in the experiments this number is usually around 5 and within 10. Consequently, *B-POL* runs fast and usually takes less than 10 seconds on a Pentium 3.2G Dual-Core CPU. Although the time cost of *B-POL* depends on the size of the unlabeled data, *B-POL* can usually run fast due to the fact that the entity features are first selected to filter out irrelevant data so that a much smaller set of unlabeled data is finally fed into *B-POL*.

**Discussion.** *PORE* aims to extract relationships between Wikipedia entities, where it can make use of the entity features in the data filtering process. Although *PORE* is not intended to extract attributes, it can be applied to extracting various relationships, which is, to some degree, reflected by the demonstrated relations selected based on the criteria mentioned previously. Moreover, the core part *B-POL* is a general learning algorithm since it is independent of the data filtering process. At present, we choose only four relations from the infoboxes for the experimentation. In the near future, we plan to apply *PORE* to many other relations which do not come from the infoboxes. We also plan to apply *B-POL* to extracting attributes from free text.

## 6 Conclusions and Future work

In this paper, we described the *Positive-Only Relation Extraction* (*PORE*) framework for relation extraction from Wikipedia text. We proposed *B-POL*, the core algorithm in *PORE*, for relation classification. It makes some extensions to a state-of-the-art positive-only learning approach built upon SVMs. Experimental results demonstrated that *B-POL* achieved significant improvements on the performance, especially when the amount of the training data is small. We also empirically showed that *B-POL* significantly outperforms the multi-class classification approach. In addition, we demonstrated the feature engineering and data filtering components of *PORE*. Although *PORE* is applied in the context of Wikipedia, the core algorithm *B-POL* is a general approach for Ontology Population and can be adapted to other domains.

In the future, we would like to investigate an optimization technique to uncover the best value of parameter *c* of *B-POL* given the positive and unlabeled data. We also plan to improve the data filtering component of *PORE* in the near future.

## References

1. Ding, L., Finin, T.: Characterizing the Semantic Web on the Web. ISWC'06.
2. Ramakrishnan, C., Kochut, K.J., Sheth, A.P.: A Framework for Schema-Driven Relationship Discovery from Unstructured text. ISWC06

3. Sergey, B.: Extracting Patterns and Relations from the World Wide Web. WebDB'98.
4. Agichtein, E., Gravano, L.: Snowball: Extracting Relations from Large Plain-text Collections. ACM DL'00.
5. Pantel, P., Pennacchiotti, M.: Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. COLING'06.
6. Ravichandran, D. and Hovy, E.H. 2002. Learning Surface Text Patterns for a Question Answering System. ACL'02.
7. Boer, V., Someren, M., Wielinga, B., J.: Extracting Instances of Relations from Web Documents using Redundancy. ESWC'06.
8. Cimiano, P., Handschuh, S., Staab, S.: Towards the Self-Annotating Web. WWW'04.
9. Mori, J., Tsujishita, T., Matsuo, Y., Ishizuka, M.: Extracting Relations in Social Networks from Web using Similarity between Collective Contexts. ISWC'06.
10. Tang, J., Hong, M., Li, J., Liang, B.: Tree-structured Conditional Random Fields for Semantic Annotation. ISWC'06.
11. Chang, C.-C., Lin, C.-J.: LIBSVM: A Library for Support Vector Machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
12. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Suder, R.: Semantic Wikipedia. WWW'06.
13. Auer, S., Lehmann, J.: What have Innsbruck and Leipzig in common? Extracting Semantics from Wiki Content. ESWC'07.
14. Yu, H., Zhai, C.X., Han, J.: Text Classification from Positive and Unlabeled Documents. CIKM'03.
15. Li, X., Liu, B.: Learning to Classify Texts Using Positive and Unlabeled Data. IJCAI'03.
16. Rocchio, J.: Relevance Feedback in Information Retrieval. In G. Salton. The smart retrieval system: experiments in automatic document processing, 1971.
17. Denoyer, L.: The Wikipedia XML Corpus. SIGIR Forum (2006)
18. Suchanek, F.M., Ifrim, G., Weikum, G.: Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents. KDD'06.
19. Chen, J., Ji, D., Tan, C.L., Niu, Z.: Relation Extraction Using Label Propagation Based Semi-supervised Learning. ACL'06.
20. Zhang, Z.: Weakly-Supervised Relation Classification for Information Extraction. CIKM'04.
21. Wang, T., Li, Y., Bontcheva, K., Cunningham, H., Wang, J.: Automatic Extraction of Hierarchical Relations from Text. ESWC'06.
22. Mitchell, T.M.: Machine Learning. McGraw-Hill Education, 2005.
23. Schölkopf, B. et al.: New Support Vector Algorithms. Neural Computation (2000)
24. Wang, G., Zhang, H., Wang, H., Yu, Y.: Enhancing Relation Extraction by Eliciting Selectional Constraint Features from Wikipedia. NLDB'07.
25. Ruiz-Casado, M., Alfonseca, E., Castells, P.: Automatic extraction of semantic relationships for WordNet by means of pattern learning from Wikipedia. NLDB'05.
26. Zhou, G.D., Su, J., Zhang, J., Zhang, M.: Exploring Various Knowledge in Relation Extraction. ACL'05.
27. Schutz, A., Buielaar, P.: RelExt: A Tool for Relation Extraction from Text in Ontology Extension. ISWC'05.
28. Manevitz, L.M., Yousef, M., editors: One-Class SVMs for Document Classification. Journal of Machine Learning Research 2 (2001) 139-154
29. Wang, G., Yu, Y., Zhu, H.: Tech. Report. Available at http://apex.sjtu.edu.cn/apex_wiki/Papers?action=AttachFile&do=get&target=wang-iswc07-tr.pdf
30. Zhu, X.: Semi-supervised Learning Literature Survey. TR 1530, Univ. of Wisconsin, Madison. Dec. 2006.
31. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. WWW'07.