Lightly-Supervised Attribute Extraction

Kedar Bellare[1], **Partha Pratim Talukdar**[2], **Giridhar Kumaran**[1], **Fernando Pereira**[2]
Mark Liberman[2], **Andrew McCallum**[1], **Mark Dredze**[2]

[1] Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
{*kedarb,giridhar,mccallum*}*@cs.umass.edu*

[2] Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{*partha,pereira,myl,mdredze*}*@cis.upenn.edu*

Abstract

Web search engines can greatly benefit from knowledge about attributes of entities present in search queries. In this paper, we introduce lightly-supervised methods for extracting entity attributes from natural language text. Using these methods, we are able to extract large numbers of attributes of different entities at fairly high precision from a large natural language corpus. We compare our methods against a previously proposed pattern-based relation extractor, showing that the new methods give considerable improvements over that baseline. We also demonstrate that query expansion using extracted attributes improves retrieval performance on underspecified information-seeking queries.

1 Attributes in Web Search

Web search engines receive numerous queries requesting information, often focused on a specific entity, such as a person, place or organization. These queries are sometimes general requests, such as *"bio of George Bush,"* or specific requests, such as *"new york mayor."* Accurately identifying the entity (*new york*) or related attributes (*mayor*) can improve search results in several ways [1]. For example, knowledge of attributes and entities can identify a query as being a factual request [1, 2]. Query expansion using known attributes of the entity can also improve results [3]. Additionally, an engine could suggest alternative queries based on attributes. If a user searches for just *"Craig Ferguson"* and *"shows"* is a known attribute of the entity *"Craig Ferguson"*, then an alternative query suggestion could be *"Craig Ferguson shows"* which may guide the user to more informative results. The widely explored technique of pseudo relevance feedback can also benefit from a known list of entities and attributes [4]. Some view entity and attribute extraction as a primary building block for the automatic creation of large scale knowledge bases aimed at addressing these issues [1].

The first step towards improving search results with attributes is to create lists of entities and attributes. Towards that end, we propose new algorithms that, beginning with a small seed set of entities and attributes, learn to extract new entities and attributes from a large corpus of text. We adopt a bootstrapping approach, where the inputs for our learning algorithms are a large unlabeled corpus and the small seed set containing an entity type of interest, such as seed pairs automatically extracted from query logs [1]. The seed pairs are matched against the corpus to create training instances for the learning algorithms. The algorithms exploit a wide range of instance features to alleviate the effects of noise and sparseness. The algorithms produce a large list of entities and associated attributes, which can be directly applied towards improving web search.

This paper proceeds as follows. We begin with some background on attribute extraction and web search applications. We then outline our extraction algorithms. Some examples and evaluations of extracted attributes and entities follow.

2 Background

Minimally supervised extraction of relation tuples has been the subject of considerable recent investigation [5, 6, 7]. These methods start from a small set of known tuples in the relation of interest, and bootstrap extractors that can find more tuples from text. For example, if the relation of interest is "*corporate acquisition*", such an extractor may learn patterns that extract the pair (*Acme Corp.*, *XYZ Inc.*) from the sentence *"XYZ Inc. was acquired by Acme Corp. for 10 million in cash.*". Previous methods [6, 8] have sought specific relationships like "*headquartered at*" or "*instance of*." In contrast, we seek to identity all commonly used attributes of an entity type, such as the *capital*, *population*, and *GDP* attributes for countries. Another important contrast to some notable previous applications of bootstrapping to named-entity recognition [9] and relation extraction [6] is that those previous efforts exploit task constraints to recognize positive instances for one case as negative instances for another case. For instance, any instance of an entity type used by [9] is a negative instance for all other types; the functional constraints required by [6] allow a positive instance xRy to be used as a negative instance of xRy' for all $y' \neq y$. In contrast, the set of possible attributes for a given entity type is open-ended, so we need some other way of eliciting negative evidence.

Some methods [1] extract entity-attribute pairs from Web query logs, while others [10] extract entity-attribute-value triples from product descriptions. The former [1] only uses a small set of linguistically motivated extraction patterns while our methods learn such patterns automatically from text, similar to earlier semi-supervised relation extraction work [5, 6, 7]. Probst et. al. [10] extract both attributes and their values from product descriptions. In contrast, we look for attributes only, because many occurrences of an entity attribute do not specify a value in the more general texts we are dealing with. The attribute extraction algorithm described by [11] has some similarities with our work. However, we use arbitrary features rather than just textual patterns during bootstrapping, we use different reliability estimation measures, and we gain accuracy from re-ranking.

Collins and Singer [9] describe a co-training based algorithm for named-entity recognition which relies on seeds from a fixed set of classes for learning. Bootstrapping approaches based on co-training and self-training have also been applied to word sense disambiguation [12], where a fixed set of classes is also used. As we noted earlier, classification into a fixed set of classes is easier to learn than attribute extraction because the mutual exclusion between classes allows positive instances for one class to be used as negative instances for other classes. Following Pantel et.al. and their Espresso system [7], we present an attribute-extraction method that uses positive instances alone and relies on mutual information estimates for extraction confidence. We also explore a more sophisticated Maximum Entropy classifier to address the attribute-extraction task. Although such classifiers have been used earlier to extract hypernymy relations from text [13] the algorithms are not applied in a bootstrap setting with few positive examples. Training data in [13] is automatically generated by looking at the WordNet hierarchy which has abundant positive and negative examples. Again, in contrast to our task, the structure of the hypernym task allows the automatic generation of negative instances, because the hypernym relation is antisymmetric.

The work presented in this paper can be considered as a generalization of the minimally supervised pattern-based relation extraction approaches previously developed by DIPRE [5] and Espresso [7], and originally pioneered by Hearst et.al.[14]. In contrast to the earlier work, our methods work on a fixed corpus without auxiliary sources, and we use generic learning algorithms with a wide range of features rather than specialized pattern-induction methods. The extraction algorithm also has a second re-ranking stage to improve accuracy over this extracted set unlike previous algorithms. Finally, as noted before, the attribute extraction task lacks constraints like a fixed set of classes, functional relationships to help generate negative instances.

3 Methods

We describe here two methods for entity-attribute extraction from a text collection. The first method (Section 3.1) learns decision lists by co-training using a mutual information-based measure. The second method (Section 3.2) learns a maximum-entropy classifier by self-training.

We first tag a given text corpus with part-of-speech information. From each tagged sentence, we extract all (proper noun, noun) pairs and consider each such pair as a candidate entity-attribute instance. We represent each such entity-attribute instance (e, a) by the set of its binary features $\mathbf{x} =$

$\{x_1, \cdots, x_k\} \subseteq \mathcal{X}$ where $\mathcal{X}$ is the set of possible features; class labels are drawn from a finite set $\mathcal{Y}$. Here, $\mathcal{Y}$ consists of only two labels $\mathcal{Y} = \{+1, -1\}$ corresponding to correct and incorrect entity-attribute pairs respectively. The seed instances are $\mathcal{D} = \{(\mathbf{x}^{(1)}, +1), (\mathbf{x}^{(2)}, +1), \ldots, (\mathbf{x}^{(k)}, +1)\}$. After training the classifiers starting from the seed examples, we produce a ranking over all instances ordered by their confidence scores. The ordered candidate set is further re-ranked using the co-training algorithm. Finally, we choose a cut-off on the number of tuples extracted to evaluate the performance of our algorithms.

3.1 Decision List Co-training

A decision list is a function $d : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]$ that maps a feature and a label to a confidence value $d(x, y)$. That is, d represents a set of decision rules $x \rightarrow y$ with weights $d(x, y)$. The decision list d can then be used to label an instance as follows:

$$\hat{y} = \arg \max_{x \in \mathbf{x}, y \in \mathcal{Y}} d(x, y)$$

That is, the instance gets the label for the decision rule in d with the highest weight.

We adapt the co-training algorithm (*DL-CoTrain*) proposed by [9] to learn a decision list (DL) given examples from only one class using co-training. The co-training algorithm splits the feature set into two views (*content* and *context*) $\mathcal{X}_1 \times \mathcal{X}_2 : \mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$. The *DL-CoTrain* algorithm induces decision lists for multiple classes starting from user-provided content seed features for each class. In our case, a class corresponds to an entity type. For each type (for instance, *country*), we provide seeds representing possible attributes of that type (for instance, $(China, area)$).

Starting from an initial seed set of content decision rules for an entity type, confidence values are estimated for context decision rules. This paper uses a confidence estimation scheme similar to the one proposed by [7]. A fixed number (n) of new rules with the highest confidence values are added to the context decision list. Once that context rules have been induced, the confidences of new content decision rules are estimated. This process is repeated until a maximum of $N_{\max}$ rules are gathered or the confidence estimates drop below a certain threshold $\tau_{\min}$. The confidence for particular features in the two views is given recursively by

$$\mathrm{C}(x_1) = \frac{\sum_{x_2 \in \mathcal{X}_2} (\frac{\mathrm{MI}(x_1, x_2)}{\mathrm{MI}_{\max}} \times \mathrm{C}(x_2))}{|\mathcal{X}_2|} \tag{1}$$

$$\mathrm{C}(x_2) = \frac{\sum_{x_1 \in \mathcal{X}_1} (\frac{\mathrm{MI}(x_1, x_2)}{\mathrm{MI}_{\max}} \times \mathrm{C}(x_1))}{|\mathcal{X}_1|} \tag{2}$$

where $\mathrm{C}(x_1)$ is the confidence of feature x_1, $\mathrm{C}(x_2)$ is the confidence of feature x_2, $\mathrm{MI}(x_1, x_2)$ is the pointwise mutual information (pmi) [15] between features x_1 and x_2. $\mathrm{MI}_{\max}$ is the maximum pmi between all features $x_1 \in \mathcal{X}_1$ and $x_2 \in \mathcal{X}_2$. It is worth noting that $\mathrm{C}(x_1)$ and $\mathrm{C}(x_2)$ are recursively defined. We initialize $\mathrm{C}(x_1) = 1.0$ for the manually supplied seed features. The pointwise mutual information is then calculated for two features $x_1 \in \mathcal{X}_1$ and $x_2 \in \mathcal{X}_2$ as follows,

$$\mathrm{MI}(x_1, x_2) \quad = \quad \mathrm{DF} \times \log \frac{|x_1, x_2||*, *|}{|x_1, *||*, x_2|}$$

where $|x_1, x_2|$, $|x_1, *|$, $|*, x_2|$ and $|*, *|$ are the counts of feature co-occurrences for x_1 with x_2, x_1 with any $x_2' \in \mathcal{X}_2$, x_2 with any from $x_1' \in \mathcal{X}_1$ and any $x_1' \in \mathcal{X}_1$ with $x_2' \in \mathcal{X}_2$ respectively. A discount factor (DF), as suggested by [8], may be used to prevent low-count features from getting a higher rank.

Finally, for an entity-attribute pair (e, a) corresponding to instance with features $\mathbf{x}$ we define the confidence score to be $c(e, a) = \max_{x \in \mathbf{x}} \mathrm{C}(x)$, where $\mathrm{C}(x)$ is given by Equations (1) and (2).

3.2 Maximum Entropy Self-training

As an alternative to decision list co-training we also present a method based on maximum entropy [16]. Given a set of entity-attribute seeds, candidate instances in the text collection that match the seeds are labeled as positive instances. The remaining candidates are regarded as negative instances.

A maximum-entropy (MaxEnt) classifier is trained on this data set using both *content* and *context* features. The classifier conditional probabilities $p(y = 1|\mathbf{x})$ are used as confidence for candidates in the unlabeled data set. The n new pairs with the highest confidence are added to the seed set.

Confidence values are used in subsequent training to set weights for the training instances, making low confidence instances play a proportionally lesser role in the trained classifier, reducing the impact of false negatives in the training set. Initially, all instances have a weight of $w = 1$. If η is the learning rate and $w_{\max}$ is the maximum instance weight, the new instance weight is set to be:

$$w_{\text{new}}^{(i)} = \min \begin{cases} w_{\max} \\ w_{\text{old}}^{(i)} + \eta \times y^{(i)} \times p(y = 1|\mathbf{x}^{(i)}) \end{cases}$$

where

$$y^{(i)} = \begin{cases} +1 & \text{if } i\text{th instance in seed set} \\ -1 & \text{otherwise} \end{cases}$$

This process is repeated for a fixed number of iterations N. Finally, the instance weights w are used to rank candidate tuples during extraction and instances with the highest confidence scores are kept. Hence, for an entity-attribute pair (e, a) with features $\mathbf{x}$ we define the confidence score to be its corresponding instance weight $c(e, a) = w$.

In our experiments, we found that the MaxEnt model was quite robust to noise, in particular to potential positive instances in the negative training set, and also to the large positive-negative imbalance. Additionally, we restricted ourselves to using few simple content and context features to avoid overfitting in the maximum entropy training.

3.3 Extraction Algorithm

Our extraction algorithm proceeds in two stages:

1. Pair extraction: At this stage, either the PMI-CoTrain or the MaxEnt algorithm are run for a specified number of iterations N to extract entity-attribute pairs. Both these methods allow us to use arbitrary instance features but care needs to be taken that the features generated are neither too specific (lower recall) nor too general (lower precision). Our experiments below suggest that this is possible.
2. Re-ranking: The previously extracted pairs are re-ranked using the score given by Equation (3) below, and the top-ranking pairs are returned.

The re-ranking score $R(e, a)$ for an entity-attribute pair is the product of the confidence score $c(e, a)$ from the first stage and the co-training scores defined by Equations (1) and (2) for the features specifying the identity of the entity and attribute:

$$R(e, a) = c(e, a) \times \mathrm{C}(\text{ent} = e) \times \mathrm{C}(\text{attr} = a) \tag{3}$$

In the above equation, the content ($\mathcal{X}_1$) and context ($\mathcal{X}_2$) features are $(\text{ent} = e)$ and $(\text{attr} = a)$ respectively. The idea behind this re-ranking is that we should have confidence in an attribute value which is strongly associated with many reliable entities. As a side-effect, the confidence values $\mathrm{C}(\text{ent} = e)$ and $\mathrm{C}(\text{attr} = a)$ allow us to rank entities and attributes for the given entity type.

4 Experimental Results

We present attribute-extraction experimental results for company attributes and country attributes. In addition, we present results on a document retrieval task where we make use of attributes for query expansion.

4.1 Baseline

The baseline attribute extractor, *BL*, is our reconstruction of the of the *Espresso* system described by [7]. We skipped the Web expansion phase of *Espresso*, and thus we do not use generic patterns, because our goal is to measure effectiveness of the algorithms on a fixed corpus. *Espresso* uses relatively complicated heuristics to select the initial set of reliable patterns. We determined that

those heuristics did not work well for our task, so we simply select the top 100 patterns in the first iteration, which performs better. We tried to match the rest of *Espresso* as closely as it was possible given the information we had on that system. We generalized sentences by replacing terminological expressions as defined by *Espresso* by the token *TR*. The *BL* method ranks all patterns using *Espresso*'s pattern reliability measure. All patterns except the top-k are discarded where k is set to the number of reliable patterns from previous iteration plus one. Relation tuples extracted by the selected patterns are scored using *Espresso*'s instance reliability measure and the highest scoring top 200 tuples are selected as reliable tuples which are used to rank patterns in the next iteration. As in *Espresso*, the iterative extraction and ranking is continued until the average pattern reliability scored decreased by more than 50% from the previous iteration. The systems used in empirical evaluation are:

BL: The baseline as described in Section 4.1 of this paper. ***PT***: PMI-CoTraining based extractor as described in Section 3.1 using only pattern and tuple identity features. After each iteration, $n = 10$ rules are induced until $N_{max} = 3000$ rules are gathered. PT_+: ***PT*** with additional surrounding context and lexical features. $PT_+ + R$: Re-ranking applied on the extractions generated by PT_+. ***ME***: MaxEnt model (Section 3.2) with context and lexical features. As there were very few positive examples, we used very few features during training to avoid overfitting. We set $\eta = 0.01$ and $w_{max} = 1.5$. As in ***PT*** we induce $n = 10$ rules per iteration. $ME + R$: Re-ranking applied to extractions from *ME*. ***SE*** $+$ ***R***: The initial seeds are first located in the corpus. Now, for each such (e, a) seed, the text connecting entity (e) with attribute (a) (or reverse if a precedes e in text) is considered a *seed-extracting (SE)* pattern. For example, if *(Google, president)* is a seed pair with corpus mention of *"Google's CEO and president"*, then *"ORG 's CEO and ATTR"* is a *seed-extracting* pattern where *ORG* and *ATTR* are non-terminals for organizations and attributes respectively. These *seed-extracting* patterns are further used to extract more tuples and re-ranking is performed on these extractions.

Features are defined on the template:

$$LxMyR$$

where L is the left context of the instance, x is the entity (attribute), M separates the entity (attribute) from the attribute (entity), y is the attribute (entity), and R is the right context. For example, in *"The population of China exceeds ..."*, $L =$ "The'', $x =$ "$population''$, $M =$ "of'', $y =$ "$China''$ and $R =$ "$exceeds \ldots''$. Features are then tests on each of L, x, M, y, and R, and their Boolean combinations.

For each rank, precision in Tables 4(a) and 4(b) specifies the percentage of correct tuples in 200 (4 runs of 50 each) randomly selected and manually evaluated *non-seed* tuples. For concreteness, here are a few correct country-attribute pairs: *(iraq, country)*, *(nepal, districts)*, *(malaysia, problems)*, *(colombia, important highway)* and here are a few incorrect ones: *(iraq, countries)*, *(namibia, orders)*, *(india, recent tour)*, *(egypt, huge fire)*. Similarly, here are a few correct company-attribute instances: *(intel, speediest chip)*, *(limited too, chief executive)*, *(caliber system inc., controller)*, *(uniroyal inc, former chairman)* and a few incorrect ones: *(milwaukee, fact)*, *(time warner, other large entertainment companies)*, *(microsoft, assumption)*, *(wal-mart stores inc., last week)*.

4.2 Attribute Extraction Task

Company Attributes Experiment: For this experiment, we use a newswire corpus of 122 million tokens with articles collected from Wall Street Journal, AFP and Xinhua News. The first 100 company names from the Fortune 500 list are used as company seeds. In addition, 12 seed attributes are manually selected, shown in Table 2. The final set of seed tuples are obtained by taking the cross product of these two sets. Rather than concentrating on the size of the seed set, we concentrate on the practicality of the seed generation process. We feel that it is acceptable to have a relatively large seed set as long as it can be cheaply obtained, such as from query logs [1].

Results for this experiment are shown in Tables 3(a) and 4(a). Variation of precision against increasing size of ranked attribute set is shown in Table 3(a). As shown in Table 4(a), the baseline *BL* and *PT* methods are outperformed by other methods (PT_+, $PT_+ + R$, *ME*, $ME + R$ and *SE* $+$ *R*). PT_+ gives a higher precision on top ranking extractions (1k) and outperforms *ME* but does worse than *ME* on larger sets (5k and 10k). With increasing size of the ranked set *SE* $+$ *R* and $ME + R$ improve over other methods.

Country-Attribute Experiment: For this experiment, we use the same corpora as the one described above. We use the 191 United Nations member countries as country seeds and manually select 8 seed attributes, shown in Table 2. Once again, the final set of seed tuples was obtained by taking cross product of these two sets. Experimental results are shown in tables 3(b) and 4(b). For this attribute extraction task, we again find that $PT_+ + R$, $ME + R$ and $SE + R$ outperform other methods. Ranked attribute precision at varying ranks is shown on Table 3(b). Precision of ranked tuples at varying ranks is shown on Table 4(b). On Table 3(b), the inferior ranking produced by $PT_+ + R$ as compared to $SE + R$ needs explanation. Since there is no initial ranking and pruning, the *SE* phase in $SE + R$ extracted around 2.8 million candidate extractions, most of which are incorrect. On the other hand, we restricted the number of PT_+ extractions to 299k. Because of the ranking in PT_+ and the limited number of candidate extractions used, most of the noisy extractions are already removed from the input of re-ranking in $PT_+ + R$. But that is not the case with the tuples fed to re-ranking in $SE + R$. Hence, re-ranking in $SE + R$ shows less confidence in an attribute occurring with many other non-country entities, thereby settling on a more accurate ranking of attributes as shown in Table 3(b). Unfortunately, re-ranking in $PT_+ + R$ does not have incorrect entities as in $SE + R$ to exploit and thereby produces an inferior ranking of attributes. However, $PT_+ + R$ ranks the entities well and since final confidence in a pair depends on confidence of PT_+ in the pair, confidence of $PT_+ + R$ in the entity and confidence of $PT_+ + R$ in the attribute (see Equation 3), pair confidence generated by $PT_+ + R$ is unaffected (see Table 4(b)) by the problem just described .

(a) $(\text{Company}, \text{Attr})$ $(PT_+ + R)$

Top non-seed attr.
chief executive officer, suit, chief executive, unit, lawsuit, part, joint venture, announcement, news, executive vice president, president, **other companies**, **others**, *software, chief financial officer, contract, strike,* **move**, *decision, company*

(b) $(\text{Country}, \text{Attr})$ $(SE + R)$

Top non-seed attr.
presidents, border, governments, foreign ministers, government, relations, **num-num victory**, **countries**, *people, ties, prime ministers, heads of state, leaders, foreign minister, democracy, prime minister, foreign investment, political asylum, trade, cooperation*

Table 1: Top twenty extracted non-seed *Company* and *Country* attributes. Errors are bold-faced.

Relation	*Key Seeds*	*Value Seeds*
(Company, Attribute)	Top 100 Fortune-500 companies	*type, headquarters, chairman, ceo, products, revenue, operating income, net income, employees, subsidiaries, website, headquarter*
(Country, Attribute)	191 UN member countries.	*capital, largest city, official language, president, area, population, gdp, currency*

Table 2: *(Company, Attribute)* and *(Country, Attribute)* seeds used in the experiments.

(a) (*Company, Attr*)

System	*Precision*			
	@10	@20	@50	@100
$PT_+ + R$	**100%**	**85%**	70%	**70%**
$ME + R$	60%	65%	**72%**	47%
$SE + R$	90%	75%	56%	40%

(b) (*Country, Attr*)

System	*Precision*			
	@10	@20	@50	@100
$PT_+ + R$	40%	65%	64%	58%
$ME + R$	**80%**	75%	80%	77%
$SE + R$	**80%**	**90%**	**88%**	**82%**

Table 3: Non-seed attribute precision at various ranks.

Discussion: In all our experiments, we find that PT_+ consistently outperforms *BL* and *PT*. This shows that compared to the case when patterns are the sole extractors (as in *BL* and *PT*), additional context features that include surrounding words help improve precision significantly. For example, for company-attribute pairs, *"⟨ATTR⟩ of ⟨COMPANY⟩"* is a low precision generic pattern as it can extract many other relations instances (*e.g. height of John*) in addition to the desired pairs. However, if you add the surrounding context *"chairman and"* to this pattern and use the concatenated feature

(a) (*Company, Attr*) tuples

System	*Precision*		
	@1k	@5k	@10k
BL	90.5%	34.5%	22.5%
PT	78%	27%	23.5%
PT_+	**96%**	54%	29%
ME	91%	**76%**	54%
$PT_+ + R$	89%	58.5%	48%
$ME + R$	75%	63%	55.5%
$SE + R$	51.5%	68%	**71%**

(b) (*Country, Attr*) tuples

System	*Precision*		
	@1k	@5k	@10k
BL	28.5%	-	-
PT	11.5%	11.5%	13%
PT_+	48%	30.5%	31%
ME	43.5%	48.5%	35.5%
$PT_+ + R$	83%	70%	67%
$ME + R$	73.5%	77.5%	**77.5%**
$SE + R$	**84.5%**	**79.5%**	77%

Table 4: Ranked non-seed tuple precision at various ranks. In case of (*C country, Attr*), *BL* extracted only 3114 non-seed tuples and hence evaluation for only top ranking 1000 non-seed extractions are shown.

(*surrctxt=chairman_and_&&_⟨ATTR⟩_of_⟨COMPANY⟩*) as an extractor then it becomes a high precision company-attribute extractor. Improvements of our methods over *BL* and *PT* in tables 4(a) and 4(b) reflect this fact. Hence, by using such novel features we are able to exploit generic patterns even without using external resources such as the Web search results used by [7]).

As the ranked set size increases, *ME* does better than PT_+because at higher ranks a few high-confidence features can distinguish between correct and incorrect entity-attribute pairs. However, using multiple features provides increased evidence for extraction at lower ranks.

By analyzing the data, we observed that company-attribute instances tend to occur in a much more repetitive context than country-attribute instances. For example, there are only 100 initial seed-extracting company-attribute patterns compared to 1259 for country-attribute pairs. Also, the country-attribute patterns appear to be less specific than the company-attribute patterns. Therefore, methods relying only on patterns such as *BL* and *PT* perform relatively better for company-attribute extraction (Table 4(a)) than for country-attribute extraction (Table 4(b)). These experiments expose the pitfalls of using pattern-only extractors and at the same time show how one can recover from them by using richer feature-based predictors.

Re-ranking seems to be more effective for country-attribute pairs than for company-attribute ones. The larger seed set may explain this.A re-ranking scheme is more likely to be useful in extraction and ranking if there is propagation of information between entities and attributes. To the best of our knowledge, we are not aware of any previous work which exploits re-ranking of this nature to improve base ranking in an attribute (or relation) extraction task.

The good performance of $SE + R$ is quite interesting, given that there is no base ranking involved. However, since there is no base ranking, $SE + R$ sometimes extracts much more noisier tuples compared to the other methods such as PT_+.

4.3 Document Retrieval Task

Templated queries are a new and emerging paradigm in search. For example, *"PROVIDE INFORMATION ON* [*organization*]*"*, is a templated query that a user interested in details about organizations can use. By instantiating the [organization] slot in the query, a user can indicate the organization she is interested in. In addition to the organization name, the user can indicate addition preferences such as date ranges, related terms and so on. Templated queries thus offer a convenient way for users to express complex information needs. On the other hand, templated queries also offer the potential to develop techniques particularly suited for retrieving specific types of information.

To measure the utility of the extracted attributes in query expansion, we performed a document retrieval task using the Indri [17] system whose results are reported in this section. Our document collection, referred to as the GALE [18] corpus, is a mix of English, Chinese, and Arabic newswire, blogs, and broadcast news. Automatic speech recognition and machine translation was used to convert sources to English text when appropriate. The entire corpus was annotated with entities identified as part of the Automatic Content Extraction (ACE) program. The presence of such disparate sources in the collection makes information retrieval challenging. Our focus was on the template

query mentioned earlier: *"PROVIDE INFORMATION ON* [*organization*]*"*. Our baseline runs were done using a simple structured query that included just the organization's name and a specification that the name should have been annotated as being of type ORG. New queries were created by combining the baseline query with the set of attributes and weighting the baseline query twenty times higher than the attributes. For example, the baseline query *#combine(hamas #1(hamas).ORG)* was converted to *#weight(1.0 #combine (members plans) 20.0 #combine (hamas #1(hamas).ORG))* to accommodate the identified attributes (e.g. "members", "plans", etc.). Performance was measured using average precision (AP) on the set of retrieved documents.

For our experiment, we found attributes of different organizations from the corpus using the methods described earlier. Note that the attributes found were on a separate corpus from the actual document collection on which retrieval is performed. We considered four query entities to fill in the *"Organization"* slot in the template query mentioned above, with each instantiation generating a separate baseline query. The four entities were: *"National Security Agency"*, *"Hamas"*, *"African Union"* and *"Liberation Tigers of Tamil Eelam"*. By adding all the attributes found by our attribute extraction method ($PT_+ + R$), the retrieval performance increases only for the entity *"Hamas"* but decreases on the other entities. As is seen in Table 5, we conjecture that since we do not have enough instances for the other three entities in the corpus from which attributes were extracted, there was not enough evidence to find good attributes for these three entities. Interestingly, the attribute extraction algorithm assigned lower confidence (in general) to the extracted attributes of these entities compared to that of *Hamas*. Attribute extraction (and binary relation extraction in general) is bi-lexical and so sparsity of data problem is more acute. Handling such cases for attribute extraction is left as future work. For the case of web search, this problem can be partially addressed by sampling a larger number of documents from which the attributes are extracted. The attributes which are extracted with high confidence can then be successfully used for expanding information-seeking under-specified queries, as shown here in the case of *"Hamas"*.

Query Entity	Baseline	Expansion	Corpus Count of Query Entity
African Union	0.2396	0.2150	164
Hamas	0.1474	0.1511	3456
Liberation Tigers of Tamil Eelam	0.5004	0.4279	406
National Security Agency	0.2382	0.2382	32

Table 5: Retrieval performance in terms of Average Precision (AP) of baseline query versus the query expanded with all attributes. The last column indicates the counts of the entities found in the corpus from which attributes were extracted.

5 Conclusion

In this paper, we have presented novel extraction and ranking mechanisms for entity and attribute extraction. The methods presented here open up the possibility of using rich features for pattern and pair reliability estimation, especially in the constrained setting where only a few positive seed instances from a single relation of interest are available, along with large unlabeled data but without any negative instances. By exploiting such rich features, our methods outperform standard pattern-based approaches which have been previously applied on attribute extraction and other relation extraction tasks. We also present a co-training-inspired re-ranking method that is very effective in the experiments reported in the paper. We also demonstrate that query expansion using the extracted attributes improves document retrieval performance on underspecified information-seeking queries.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by DARPA under contract number HR0011-06-C-0023, in part by the Central Intelligence Agency, in part by the National Security Agency and by NSF grants EIA-0205448, IIS-0326249. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

[1] Marius Pasca and Benjamin Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI-07. Ferbruary, 2007*, 2007.

[2] X. Li and D. Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics*, 2002.

[3] Marius Pasca. Organizing and searching the world wide web of facts - step two: Harnessing the wisdom of the crowds. In *16th Internation World Wide Web Conference, May 2007*, 2007.

[4] Luiz Augusto Pizzato, Diego Mollá, and Cécile Paris. Pseudo-relevance feedback using named entities for question answering. In *Australasian Language Technology Workshop (ALTW2006)*, pages 83–90, 2006.

[5] Sergey Brin. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, 1998.

[6] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *5th ACM International Conference on Digital Libraries*, 2000.

[7] Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *COLING/ACL-06, Sydney, Australia*, 2006.

[8] Patrick Pantel and Deepak Ravichandran. Automatically labeling semantic classes. In *HLT/NAACL-04, Boston, MA*, 2004.

[9] Micahel Collins and Yoram Singer. Unsupervised models for named entity classification. In *In Proceedings of EMNLP-1999*, 1999.

[10] Katharina Probst, Rayid Ghani, Marko Krema, Andy Fano, and Yan Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *IJCAI-07, Ferbruary, 2007*, 2007.

[11] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *EMNLP 2005*, 2005.

[12] Rada Mihalcea. Co-training and self-training for word sense disambiguation. In *CoNLL*, 2004.

[13] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2005*, 2005.

[14] Marti Hearst. Automatic acquisition of hyponyms from large text corpora. In *Fourteenth International Conference on Computational Linguistics, Nantes, France*, 1992.

[15] T.M. Cover and J.A. Thomas. Elements of information theory. In *John Wiley & Sons*, 1991.

[16] Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[17] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*, May 2-6, 2005.

[18] Giridhar Kumaran and James Allan. Information Retrieval Techniques for Templated Queries. In *Proceedings of RIAO 2007 (Recherche d'Information Assistee par Ordinateur - Computer assisted information retrieval)*, 2007.