Learning Text Patterns for Web Information Extraction and Assessment (Extended Version)

Doug Downey, Oren Etzioni, Stephen Soderland, and **Daniel S. Weld**

Department of Computer Science and Engineering
University of Washington
Seattle, WA-98195
ddowney@cs.washington.edu, etzioni@cs.washington.edu, soderlan@cs.washington.edu, weld@cs.washington.edu

Abstract

Learning text patterns that suggest a desired type of information is a common strategy for extracting information from unstructured text on the Web. In this paper, we introduce the idea that learned patterns can be used as both extractors (to generate new information) and discriminators (to assess the truth of extracted information). We demonstrate experimentally that a Web information extraction system (KnowItAll) can be improved (in terms of coverage *and* accuracy) through the addition of a simple pattern-learning algorithm. By using learned patterns as extractors, we are able to boost recall by 50% to 80%; and by using such patterns as discriminators we are able to reduce classification errors by 28% to 35%. In addition, the paper reports theoretical results on optimally selecting and ordering discriminators, and shows that this theory yields a heuristic that further reduces classification errors by an additional 19% to 35% – giving an overall error reduction of 47% to 53%.

1. Introduction

A variety of recent work aimed at extracting information from free text uses a form of *pattern learning* (e.g. Soderland 1999; Riloff & Jones 1999; Lin, Yangarber, & Grishman 2003; Ravichandran & Hovy 2002). Starting with a set of seed examples of a given class, pattern learning algorithms scan a corpus to discover contextual patterns in which instances of the class are commonly found. The discovered patterns can then be used on the corpus as *extractors* to generate instances of the class. When pattern learning is applied to a large corpus (like the Web), the automatic creation of large knowledge bases becomes an exciting possibility (e.g. Agichtein & Gravano 2000; Brin 1998).

A common problem with information extraction systems is that the quality of the extracted information is variable and can degrade as extraction progresses. Inspired by Turney's PMI-IR algorithm (Turney 2001), our recent work addressed this problem by using patterns as *discriminators* (Etzioni et. al 2004a). We gather hit counts from Web search engines to compute the *pointwise mutual information* (PMI) between an extraction and a discriminator for the class, and we use this "web-scale" statistic as an independent assessment of the veracity of the extraction. For example, the hit count of the phrase "Boston and other cities" can be used to estimate the probability that "Boston" is in fact a city.

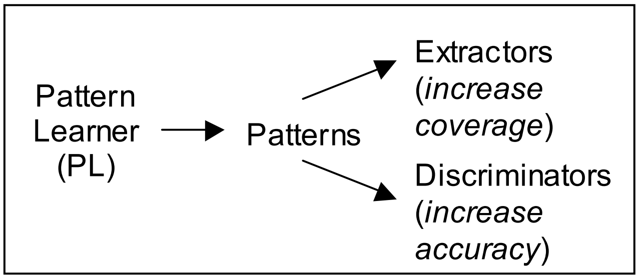
Pattern Learner (PL)
Patterns
Extractors (*increase coverage*)
Discriminators (*increase accuracy*)

Figure 1: The patterns that PL produces can be used as both extractors and discriminators.

In this paper, we investigate applying a simple pattern learning algorithm (PL) to the task of Web information extraction. Our primary contributions are:

1. As shown in Figure 1, we introduce the insight that PL can be used increase *both* coverage (by learning extractors) and accuracy (by learning discriminators).
2. We quantify the efficacy of this approach via experiments on multiple classes, and describe design decisions that enhance the performance of pattern learning over the Web.
3. We introduce a theoretical model of discriminator ordering and selection and show that, while the general problem is NP-hard, ordering discriminators by *Marginal Utility (MU)* is optimal in important special cases. As suggested by the theory, *MU* is shown to be effective at increasing accuracy in practice.

We use KnowItAll (Etzioni et. al 2004a), a Web information extraction system, as a baseline for our experiments. The baseline KnowItAll system does not rely on pattern learning; it instead uses a set of *domain independent patterns* (*cf.* Hearst 1991) as both extractors and discriminators. For example, the generic pattern "NP1 such as NP2" indicates that the head of the noun phrase in NP2 is a member of the class named in NP1. Instantiated

Technical Report UW-CSE-04-05-01
May 2004

for different classes (e.g. producing the pattern "cities such as <*City*>") these patterns have been successful in generating large, high accuracy collections of facts from the Web. The experiments in this paper compare the baseline KnowItAll system with an enhanced version that includes learned patterns in addition to domain independent patterns.

The method we use to learn patterns is described in Section 2. We then describe our experience using learned patterns as extractors (Section 3) and as discriminators (Section 4). Related work is discussed in Section 5, and we conclude with directions for future work in Section 6.

2. Learning Patterns

Our pattern learning algorithm (PL) proceeds as follows:

(1) Start with a set *I* of seed instances generated by domain-independent extractors.

(2) For each seed instance *i* in *I*:

  Issue a query to a Web search engine for *i,* and for each occurrence of *i* in the returned documents record a *context string* – comprised of the *w* words before *i*, a placeholder for the class instance (denoted by "<*class-name*>"), and the *w* words after *i*. (Here, we use *w* = 4).

(3) Output the best *patterns* according to some metric – a *pattern* is defined as any substring of a context string that includes the instance placeholder and at least one other word.

The goal of PL is to find high-quality patterns. A pattern's quality is given by its *recall* (the fraction of instances of the target class that can be found on the Web surrounded by the given pattern text) and its *precision* (the fraction of strings found surrounded by the pattern text that are of the target class). The Web contains a large number of candidate patterns (for example, PL found over 300,000 patterns for the class City), most of which are of poor quality. Thus, estimating the precision and recall of patterns efficiently (i.e. without searching the Web for each candidate pattern) is important. Estimating precision for patterns is especially difficult because we have no labeled negative examples, only positive seeds. Instead, in a manner similar to (Lin, Yangarber, & Grishman 2003) we exploit the fact that PL learns patterns for multiple classes at once, and take the positive examples of one class to be negative examples for all other classes. Given that a pattern *p* is found for *c*(*p*) distinct seeds from the target class and *n*(*p*) distinct seeds from other classes, we define:

$$EstimatedPrecision(p) = \frac{c(p)+k}{c(p)+n(p)+m} \quad (1)$$

$$EstimatedRecall(p) = \frac{c(p)}{S} \quad (2)$$

where *S* is the total number of seeds in the target class, and *k*/*m* is a constant prior estimate of precision, used to perform a Laplace correction in (1). This prior estimate was chosen based on testing extractions from a sample of the learned patterns using PMI Assessment.

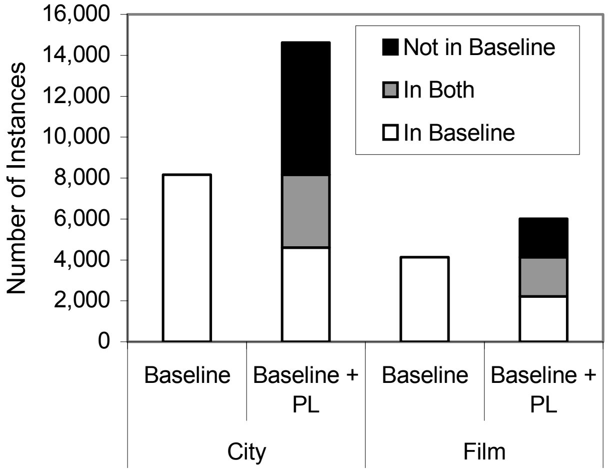

Figure 2: Unique instances of City and Film at precision 0.9. Pattern learning increases coverage by 50% to 80% over the baseline.

3. Learned Patterns As Extractors

The patterns PL produces can be used as extractors to search the Web for new candidate facts. For example, given the learned pattern "headquartered in <*City*>," we search the Web for pages containing the phrase "headquartered in". Any proper noun phrase occurring directly after "headquartered in" in the returned documents becomes a new candidate extraction for the class City.

Of the many patterns PL finds for a given class, we choose as extractors those patterns most able to efficiently generate new extractions with high precision. The patterns we select must have high precision, and extractor *efficiency* (the number of unique instances produced per search engine query) is also important.

For a given class, we first select the top patterns according to the following heuristics:

H1: As in (Brin, 1998), we prefer patterns that appear for multiple distinct seeds. By banning all patterns found for just a single seed (i.e. requiring that *EstimatedRecall* > 1/*S* in Equation 2), 96% of the potential rules are eliminated. In experiments with the class City, H1 was found to improve the average efficiency of the resulting patterns by a factor of five.

H2: We sort the remaining patterns according to their *EstimatedPrecision* (Equation 1)[1]. On experiments with the class City, ranking by H2 was found to further increase average efficiency (by 64% over H1) and significantly improve average precision (from 0.32 to 0.58).

Of all the patterns PL generates for a given class, we take the 200 patterns that satisfy H1 and are ranked most

[1] In the case of ties in *EstimatedPrecision*, we assume that longer patterns are more precise, similar to (Brin, 1998).

Extractor Pattern	Correct Extractions	Precision
the cities of <*City*>	5215	0.80
headquartered in <*City*>	4837	0.79
for the city of <*City*>	3138	0.79
in the movie <*Film*>	1841	0.61
<*Film*> the movie starring	957	0.64
movie review of <*Film*>	860	0.64

Table 1: Three of the most productive extractors for City and Film, along with the number of different correct extractions produced by each extractor, and the extractor's overall precision (before assessment).

highly by H2 and subject them to further analysis, applying each to 100 Web pages and testing precision using PMI assessment.

Results

We performed experiments testing our Baseline system (KnowItAll with only domain independent patterns) against an enhanced version, Baseline+PL (KnowItAll including extractors generated by pattern learning). In both configurations, we perform PMI assessment to assign a probability to each extraction (using only domain independent discriminators). We estimated the *coverage* (number of unique instances extracted) for both configurations by manually tagging a representative sample of the extracted instances, grouped by probability. In the case of City, we also automatically marked instances as correct if they appeared in the Tipster Gazetteer. To ensure a fair comparison, we compare coverage at the same level of overall precision*,* computed as the proportion of correct instances at or above a given probability.

The results shown in Figure 2 show that using learned patterns as extractors improves KnowItAll's coverage substantially, by 50% to 80% (we choose precision level 0.9 as representative of high-quality extraction, although the results are qualitatively similar for precision levels between 0.80 and 0.95). Examples of the most productive extractors for each class are shown in Table 1.

4. Patterns As Discriminators

Learned patterns can also be used as discriminators to perform PMI assessment. Any pattern *D* can be used as a discriminator on extraction *E* by computing the *PMI* of *D* and *E* defined as:

$$PMI(D,E) = \frac{\text{Hits}(D+E)}{\text{Hits}(E)} \tag{3}$$

where *D* + *E* is the discriminator pattern with the extraction substituted for the instance placeholder. For example, ("city of <*City*>" + "Chicago") indicates the phrase "city of Chicago".

The PMI scores for a given extraction are then used as features in a Naïve Bayes classifier. In the experiments below, we show that learned discriminators provide stronger features than domain independent discriminators for this classifier, improving the *classification accuracy* (the percentage of extractions classified correctly) of the PMI assessment.

Once we have a large set of learned discriminators, determining which discriminators are the "best" in terms of their impact on classification accuracy becomes especially important, as we have limited access to Web search engines. In the baseline KnowItAll system, the same five discriminators are executed on every extraction. However, it may be the case that a discriminator will perform better on some extractions than it does on others. For example, the discriminator "cities such as <*City*>" has high precision, but appears only rarely on the Web. While a PMI score of 1/100,000 on "cities such as <*City*>" may give strong evidence that an extraction is indeed a city, if the city itself appears only a few thousand times on the Web, the probability of the discriminator returning a false zero is high. For these rare extractions, choosing a more prevalent discriminator (albeit one with lower precision) like "<*City*> hotels" might offer better performance. Lastly, executing five discriminators on every extraction is not always the best choice. For example, if the first few discriminators executed on an extraction have high precision and return true, the system's resources would be better spent assessing other extractions, the truth of which is less certain.

Below, we express the problem of choosing which discriminators to execute on which extractions as an optimization problem, and give a heuristic method that includes the enhancements mentioned above. We show that the heuristic has provably optimal behavior in important special cases, and then verify experimentally that the heuristic improves accuracy.

The Discriminator Ordering Problem

We define the *discriminator ordering problem* as an optimization problem in which the goal is to obtain an accurate assessment of the probabilities of a given set of extractions using a limited number of resources. Specifically, the problem is defined by a set of extractions $\Phi = \{\phi_1, ..., \phi_M\}$, and a set of discriminators $\Delta = \{\delta_1, ..., \delta_N\}$. We assume that the precision and recall of each discriminator are known. The system can apply a given discriminator to any extraction – we define this set of possible actions as $A = \{\delta_i(\phi_j)\}$ for all $\delta_i \in \Delta,\ \phi_j \in \Phi$. Each action can be performed at most once. Executing an action $\delta_i(\phi_j)$ returns to the system a binary assessment (*true* or *false*) of the truth of the extraction ϕ_j. Also, each action has a cost $c(\delta_i(\phi_j))$.

We denote the system's current belief in extraction ϕ_i by $b(\phi_i) \in [0,1]$, where $b(\phi_i)$ is the system's estimate of the probability that ϕ_i is true. After executing an action

$\delta_i(\phi_j)$, the system changes its belief in ϕ_j using a Naïve Bayes update; we assume that the outcomes of actions $\delta_i(\phi_j)$ are conditionally independent given the actual truth-value of ϕ_j. The goal of the system is to choose actions in such a way that its final beliefs correspond as accurately as possible to the actual state of the world. Specifically, the reward function to be maximized is

$$R = \sum_{true\ \phi} b(\phi) - \beta \sum_{false\ \phi} b(\phi) \tag{4}$$

where β is a penalty factor for falsely asserting that an extraction is true. As mentioned above, each action has a cost associated with it, and the system's goal is to maximize R subject to a cost constraint C. Because transitions between states are probabilistic, the optimization problem is to find a *policy* mapping belief states to actions that maximizes the total *expected* reward $E[R]$ at cost less than or equal to C.

The discriminator ordering problem is identical to the problem of *active classification,* in which an object to be classified has a set of unknown attributes, each of which can be obtained by performing tests of varying costs (Heckerman, Breese, & Rommelse 1994; Turney 2000). In a similar formulation also using a Naïve Bayes classifier, (Guo 2002) shows that the problem of finding an optimal policy for the special case of classifying a *single* object can be mapped onto a partially-observable Markov Decision Process (POMDP). Finding optimal policies for POMDPs is known to be PSPACE-complete (Papadimitriou & Tsitsiklis, 1987); however, the particular POMDPs produced by Guo's mapping have special structure, and Guo asks if polynomial time algorithms for this particular problem may exist. However, below we state that in fact the single-object classification task is NP-hard – we then detail assumptions relevant to KnowItAll that allow the construction of a provably optimal policy.

In the following we reason about two subproblems of the discriminator ordering problem – the *single-extraction, multiple discriminator* ordering problem (the original problem with $|\Phi| = 1$) and the *multiple extraction, single-discriminator* ordering problem (the original problem with $|\Delta| = 1$).

Theorem 1: The single-extraction, multiple discriminator ordering problem is NP-hard in the number of discriminators.

Proof[2]: Given an instance of the knapsack optimization problem with a set of items I (each having an integral value v_i and a weight t_i) and knapsack weight limit L, the goal is to find a subset K of I such that the sum of the values of the items in K is as large as possible, given that the total weight of the items in K is less than L.

The reduction from the knapsack problem to the single-extraction discriminator ordering problem is straightforward, except for one technical detail – in the knapsack problem, rewards (i.e. values) are additive; in discriminator ordering, the expected reward from executing a particular discriminator can change depending on the results of other discriminator executions. We solve this by reducing knapsack to a discriminator ordering problem where discriminators have precision equal to one (so as soon as one discriminator returns true, the system knows the extraction is true and reward R is one)[3] and discriminator recall is proportional to value but small enough that the additive property of knapsack is preserved. Specifically, we reduce an instance of the knapsack problem to the single-fact discriminator ordering problem as follows: first, set $\Phi = \{\phi_1\}$ and $\beta = 1$, and set the initial belief $b(\phi_1) = \frac{1}{2}$ (when computing expectations, we take belief as our estimate of probability, so the probability that ϕ_1 is true is $P(\phi_1) = b(\phi_1) = \frac{1}{2}$). For each item i in I, we create a discriminator δ_i with cost $c(\delta_i(\phi_1)) = t_i$, precision $\pi(\delta_i) = 1$, and recall $\omega(\delta_i) = \dfrac{v_i}{2V^2|I|^2}$, where $V = \max(v_i)$. Note that the reduction to this single-extraction discriminator ordering problem is polynomial-time.

The discriminator ordering problem returns a *policy* mapping states to actions; however, with $\pi(\delta_i) = 1$ for all i, this policy amounts to an optimal set K of discriminators that has total cost < L and maximizes expected reward $E[R]$ for the system. The system executes the discriminators from K, and if one returns true, the extraction must be true. In this case the system assigns belief $B_{true} = 1$ and receives reward R = 1. If no discriminator returns true, the system will assign a belief value of B_{false} based on a Naïve Bayes update:

$$\begin{aligned} B_{false} &= P(\phi_1 \mid \{\neg\delta_i : i \in K\}) \\ &= \frac{P(\{\neg\delta_i : i \in K\} \mid \phi_1)P(\phi_1)}{P(\{\neg\delta_i : i \in K\} \mid \neg\phi_1)P(\neg\phi_1) + P(\{\neg\delta_i : i \in K\} \mid \phi_1)P(\phi_1)} \\ &= \frac{\prod_{i \in K}(1 - \omega_i(\delta_i))}{1 + \prod_{i \in K}(1 - \omega_i(\delta_i))} \end{aligned}$$

[2] Although we give the proof in terms of our active classification model, it can also be directly adapted to the POMDP model given in (Guo 2002).

[3] Note that this is a significantly simplified version of the discriminator ordering problem; in particular, it includes assumption **A2** described below.

The expected reward of executing the set of discriminators *K* is

$$E[R]_K = P(\phi_1)\begin{bmatrix}(1-P(\{\neg\delta_i : i \in K\} \mid \phi_1))B_{true} \\ +P(\{\neg\delta_i : i \in K\} \mid \phi_1)B_{false}\end{bmatrix}$$
$$-P(\neg\phi_1)B_{false}$$

With algebra we have:

$$E[R]_K = \frac{1-\prod_{i\in K}(1-\omega(\delta_i))}{2+2\prod_{i\in K}(1-\omega(\delta_i))} \qquad (5)$$

We will prove the reduction by showing that whenever *K* is a higher-value set of knapsack items than *K'* i.e.

$$\sum_{i\in K}\omega(\delta_i) > \sum_{j\in K'}\omega(\delta_j)$$

it must be the case that

$$\prod_{i\in K}(1-\omega(\delta_i)) < \prod_{i\in K'}(1-\omega(\delta_i)),$$

which by Equation 5 implies that $E[R]_K > E[R]_{K'}$.

Because the $\omega(\delta_i)$'s are sufficiently small, we can expand the product $\prod_{i\in K}(1-\omega(\delta_i))$ and bound the sum of the terms including more than one $\omega(\delta_i)$. Specifically,

$$\prod_{i\in K}(1-\omega(\delta_i)) =$$
$$1-\sum_{i\in K}\omega(\delta_i)+\sum_{i\in K}\sum_{j\in K-\{i\}}\omega(\delta_i)\omega(\delta_j)-\ldots+(-1)^{|K|}\prod_{i\in K}\omega(\delta_i)$$

It can be shown that the terms of this telescoping series are decreasing in absolute value, so that

$$\prod_{i\in K}(1-\omega(\delta_i)) = 1+Q(K)-\sum_{i\in K}\omega(\delta_i)$$

with

$$|Q(K)| \le \binom{|K|}{2}\omega_{\max}^2 < \frac{|I|^2}{2}\left(\frac{V}{2V^2|I|^2}\right)^2 = \frac{1}{8V^2|I|^2}$$

Note that this is true for all *K*, so $|Q(K')|$ is also less than $\frac{1}{8V^2|I|^2}$. Since the values v_i are positive integers, if $\sum_{i\in K}\omega(\delta_i) > \sum_{j\in K'}\omega(\delta_j)$, the difference between $\sum_{i\in K}\omega(\delta_i)$ and $\sum_{j\in K'}\omega(\delta_j)$ must be at least $\frac{1}{2V^2|I|^2}$. So in particular:

$$-\frac{1}{8V^2|I|^2}+\sum_{i\in K}\omega(\delta_i) > \frac{1}{8V^2|I|^2}+\sum_{j\in K'}\omega(\delta_j)$$
$$\Rightarrow 1+Q(K)-\sum_{i\in K}\omega(\delta_i) < 1+Q(K')-\sum_{j\in K'}\omega(\delta_j)$$
$$\Rightarrow \prod_{i\in K}(1-\omega(\delta_i)) < \prod_{i\in K'}(1-\omega(\delta_i))$$
$$\Rightarrow E[R]_K > E[R]_{K'}$$

We have shown that whenever $\sum_{i\in K}\omega(\delta_i) > \sum_{j\in K'}\omega(\delta_j)$ it must be the case that $E[R]_K > E[R]_{K'}$. Thus, a solution to the discriminator ordering problem is a set of discriminators *K* for which the total cost is less than *L*, and the sum $\sum_{i\in K}\omega(\delta_i)$ is no less than $\sum_{i\in K'}\omega(\delta_i)$ for any other *K'* with total cost less than *L*. Because $\omega(\delta_i)$ is proportional to v_i, the set *K* is also an optimal solution to the knapsack problem, completing the reduction. ■

As a corollary to Theorem 1, the general discriminator ordering problem is NP-hard.

The *MU* Heuristic

We have stated that the discriminator ordering problem is NP hard for even a single extraction. Here we define the *MU heuristic,* a policy that always chooses as the next action the one with highest expected marginal utility (*MU*), and we state conditions under which it gives provably optimal performance – the *MU* heuristic was shown to have similar properties for a different problem in (Etzioni 1991).

Definition: The *expected marginal utility* (*MU*) of applying a discriminator $\delta_i \in \Delta$ to an extraction $\phi_j \in \Phi$ is defined as the expected increase in reward, *R*, as a result of $\delta_i(\phi_j)$, divided by the cost $c(\delta_i(\phi_j))$. We can compute $MU(\delta_i(\phi_j))$ given the precision and recall of δ_i and the current belief $b(\phi_j)$:

$$MU(\delta_i(\phi_j)) = \frac{1}{c(\delta_i(\phi_j))}\begin{pmatrix}P(\delta_i(\phi_j),\phi_j)(b^+) \\ -\beta P(\neg\delta_i(\phi_j),\neg\phi_j)(b^-) \\ -\beta P(\delta_i(\phi_j),\neg\phi_j)(b^+) \\ +P(\neg\delta_i(\phi_j),\phi_j)(b^-)\end{pmatrix} \qquad (6)$$

where b^+ (resp. b^-) stands for the change in the belief value of ϕ_j when $\delta_i(\phi_j)$ returns true (false), and $P(\delta_i(\phi_j),\neg\phi_j)$, for example, is the probability that the discrimination action $\delta_i(\phi_j)$ returns true but the extraction ϕ_j is in fact false. The probabilities used to compute *MU* can be obtained for discriminator δ_i and extraction ϕ_j using the discriminator's recall $P(\delta_i(\phi_j) \mid \phi_j)$ and precision $P(\phi_j \mid \delta_i(\phi_j))$ along with the belief in the extraction $b(\phi_j)$, which as an estimate of $P(\phi_j)$.

MU achieves the enhancements in choosing discriminators mentioned above by being *extraction-sensitive* in two ways. First, as the system becomes more certain of the classification of ϕ_j (i.e. belief approaches zero or one), it can be shown that $MU(\delta_i(\phi_j))$ tends to decrease – that is, *MU* prioritizes uncertain extractions. Secondly, when computing *MU* we can use the hit count of ϕ_j to adjust the expected outcome of $\delta_i(\phi_j)$. This allows *MU* to account for the fact that rare extractions, even if true, are likely to have a PMI of zero for discriminators that also appear rarely. As an example of how this is implemented, take the case of estimating a discriminator's recall by testing the discriminator δ_i on a set of positive seed extractions. For extractions that are not rare, the portion of positive seeds for which δ_i returns "true" (i.e. those ϕ_j for which $PMI(\delta_i, \phi_j)$ is above a threshold τ set based on seed data) is taken to be the discriminator's recall, or $P(\delta_i(\phi) \mid \phi)$ for a random ϕ. However, for rare extractions, this estimate can be inaccurate. Note that for an extraction ϕ_k, the smallest non-zero PMI score possible is $1/\text{Hits}(\phi_k)$; this is obtained when $\text{Hits}(\delta_i + \phi_k) = 1$. If ϕ_k is rare and δ_i also appears rarely, this minimal positive PMI score can be much larger than τ. In this case, it is likely (even if ϕ_k is true) that the PMI between δ_i and ϕ_k will fail to exceed the threshold τ by the large margin represented by a PMI score of $1/\text{Hits}(\phi_k)$. In other words, often we would expect to observe a hit count of zero for $\delta_i + \phi_k$, even if ϕ_k is true. Thus, naively estimating $P(\delta_i(\phi) \mid \phi)$ based on non-rare seeds with PMIs below $1/\text{Hits}(\phi_k)$ tends to overstate the actual recall $P(\delta_i(\phi_k) \mid \phi_k)$ of the discriminator on ϕ_k. The heuristic solution to this problem we use here is to compute recall based on the PMI scores for seeds ϕ_j we would expect *if* the seeds had the same hit count as ϕ_k. In particular, for those ϕ_j with PMIs less than $1/\text{Hits}(\phi_k)$, we assume that if ϕ_j had the same hit count as ϕ_k then the discriminator would return a non-zero hit count a fraction of the time equal to $\text{Hits}(\phi_k) PMI(\phi_j, \delta_i)$. In this way we obtain an estimate for $P(\delta_i(\phi_k) \mid \phi_k)$ that takes the rarity of ϕ_k into account.

There are two assumptions that make ordering discriminators in KnowItAll simpler than the general formulation of the discriminator ordering problem. First, applying a discriminator currently requires issuing a single query to a Web search engine (assuming that the hit count of the extraction itself is known); thus, the cost of all actions is the same. We formalize this assumption as:

A1: The cost $c(\delta_i(\phi_j)) = 1$ for all $\delta_i \in \Delta$, $\phi_j \in \Phi$.

This assumption allows us to make the following theoretical guarantee:

Theorem 2: Given assumption **A1**, the *MU* heuristic is optimal for the multiple-extraction, single-discriminator ordering problem.

Proof: Given that the system is in a state with consumed cost *c,* let *C'* be the greatest integer less than the remaining cost *C* – *c*. Since all actions have unit cost (by **A1**), in this case the system can only choose a set of *C'* extractions on which to execute the discriminator. The expected reward of this set of actions is equal to the sum of the expected reward of each individual action, and this sum is maximized when the expected reward of each of the *C'* individual actions is as large as possible. By **A1**, the expected reward of an action is equal to its *MU* value, so the optimal policy is to execute the *C'* actions with maximal *MU.* ■

A further assumption comes from the fact that the discriminators PL finds often dominate one another for a given extraction; that is, if one discriminator has higher *MU* than another for an extraction at some belief level, it will tend to have higher *MU* for that extraction at other belief levels. Formally:

A2: If $MU(\delta_i(\phi_k)) > MU(\delta_j(\phi_k))$ when $b(\phi_k) = h$, then for all $h' = b(\phi_k)$, $MU(\delta_i(\phi_k)) > MU(\delta_j(\phi_k))$.

Theorem 3: Given assumptions **A1** and **A2,** the *MU* heuristic is optimal for the single-extraction, multiple-discriminator ordering problem.

Proof: Define the current state as follows: the belief in fact ϕ_1 is *b,* there is a set *D* of discriminators that have yet to be executed on ϕ_1, and a total cost of *c* has been consumed so far. Let *C'* be the greatest integer less than the remaining cost *C* – *c*. Since all actions have unit cost (by **A1**), the system can execute a total of *C'* discriminators. We will prove that the optimal policy is always to execute the *C'* discriminators from *D* with highest *MU* (*and* that these *C'* discriminators are always the same irrespective of both the outcome of previously executed discriminators and the initial belief *b* in fact ϕ_1) by induction on *C'*.

For the base case, note that by **A1**, the *MU* of an action is equal to the expected reward of executing that action. So for *C'* = 1, the action with maximal *MU* is exactly the action with maximal expected reward, i.e. the optimal action. It remains to prove that the action with maximal *MU* is always the same irrespective of both the outcome of previously executed discriminators and the initial belief *b* in fact ϕ_1. By the Naïve Bayes assumption, the optimal action is the same for any *given* belief *b* and does not vary

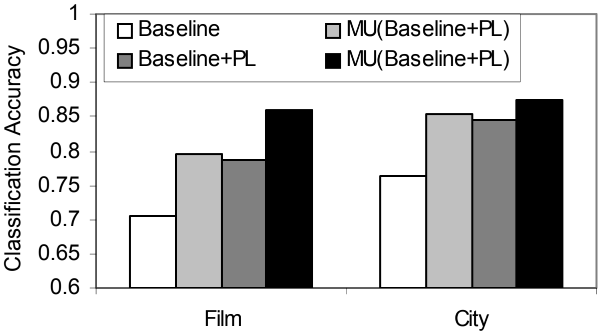

Figure 3: Classification accuracy for the classes Film and City. Adding pattern learning (PL) always improves accuracy, and ordering by marginal utility (MU) is always better than the baseline ordering (precision). The performance difference between the best method (MU(Baseline+PL)) and the Baseline method is statistically significant for each class (p < 0.01, chi-square test).

for different particular outcomes of previous discriminators; further, by assumption **A2**, the optimal action does not depend on *b*.

Now take $C' = n + 1$. Let δ be the first discriminator executed by the optimal policy. By the inductive hypothesis, after executing δ the optimal policy is to execute a set *G* containing the *n* remaining discriminators with greatest *MU* value (*and* we need only consider one such set *G*, because *G* is the same irrespective of the outcome of δ). However, the expected reward of executing δ followed by *G* is the same as that of executing *G* followed by δ. This implies that executing G followed by δ is *also* an optimal policy. However, if that is the case (by the reasoning for the base case given above) there must be no discriminator in the set *D* – *G* with higher *MU* than δ. Thus, the set of discriminators $G \cup \{\delta\}$ executed by the optimal policy is in fact the *n* + 1 discriminators in *D* with the highest *MU* value. ■

Given assumptions **A1** and **A2**, we have shown that the *MU* heuristic offers provably optimal performance in the multiple-extraction, single-discriminator and single-extraction, multiple-discriminator cases. However, in general we are interested in the multiple-extraction, multiple-discriminator case (the *MU* heuristic can be shown to give suboptimal performance in a case with at least two extractions and two discriminators). Also, the assumption **A2** is true often, but not always. Given the strong assumptions needed to prove its optimality, we experimentally validate the performance of the *MU* heuristic below.

Experimental Results

We tested the performance of PMI assessment under four different configurations – using either domain independent patterns ("Baseline") or learned patterns ("Baseline+PL") as discriminators, and ordering the execution of discriminators by either *MU* (with $\beta = 1$) or a baseline measure (precision). In each configuration, we run a total of 600 discriminators on a set of 300 extractions from each of the classes Film and City (drawn from the experiments with learned rules in Section 3). The baseline ordering always executes the same two discriminators on each extraction. The *MU* ordering, by contrast, dynamically chooses the discriminator and extraction with highest expected marginal utility, and may choose to execute more discriminators on some extractions and one or zero on others. Because our theory assumes the precision and recall of the discriminators are known, for this experiment we estimate these quantities using a hand-labeled training set (disjoint from the test set) of 100 extractions.

We used one training and test set to choose settings that maximized performance for each of our methods; we then evaluated the methods with three cross-validation runs. The average results of these cross-validation runs are shown in Figure 3**.** Adding pattern learning (PL) always improves accuracy, and ordering by marginal utility (MU) is always better than the baseline ordering. When ordered properly, the domain-independent discriminators perform especially well on the class City*,* mostly due to the particularly useful discriminator "cities <*City*>."

We explored two heuristics to ensure that our discriminators are as conditionally independent as possible, as assumed by our Naïve Bayes Classifier. We tried requiring that no two discriminator phrases executed on the same extraction are substrings of each other, and also that left- and right-handed discriminators alternate for any given extraction (a left-handed discriminator is one in which the instance placeholder appears at the right side of the pattern, with text on the left, e.g. "the film <*Film*>"). Adding both of these enhancements reduced error by an average of 9% for the *MU*-ordered configurations. The heuristics decreased performance slightly for the precision-ordered configurations when executing only two discriminators per fact (and are therefore not employed for those configurations in Figure 3).

While we have chosen the metric of classification accuracy to compare our methods, it represents only one point on the precision/recall curve, and performance at different precision/recall points can vary. In particular, ordering discriminators by precision tends to give superior performance at lower levels of recall. Also, as we would expect, *MU* offers the most benefit in cases where resources (i.e. discriminators executed) are scarce. If we increase the number of discriminators to 1200, ordering by *MU* offers only a small accuracy benefit (however, for this data set, increasing the number of discriminators to 1200 does not increase maximum achievable accuracy). Finally, the increase in accuracy found by ordering discriminators with *MU* as opposed to precision suggests that other metrics combining recall and precision would also perform well. Indeed, in experiments similar to those in Figure 3, we have found that ordering discriminators by their F-measure (with beta=1) results in accuracy closer to that of *MU* (although MU(Baseline+PL) still provides a 30% error reduction over the F-measure ordering on the Film class).

5. Related Work

PL is similar to existing approaches to pattern learning, the primary distinction being that we use learned patterns to perform PMI-IR (Turney 2001) assessment as well as extraction. PL also differs from other pattern learning algorithms in some details. (Riloff & Jones 1999) uses *bootstrapped learning* on a small corpus to alternately learn instances of large semantic classes and patterns that can generate more instances; similar bootstrapping approaches that use larger corpora include Snowball (Agichtein & Gravano 2000) and DIPRE (Brin 1998). Our work is similar to these approaches, but differs in that PL does not use bootstrapping (it learns its patterns once from an initial set of seeds) and uses somewhat different heuristics for pattern quality. Like our work, (Ravichandran & Hovy 2002) use Web search engines to find patterns surrounding seed values. However, their goal is to support *question answering*, for which a training set of question and answer pairs is known. Unlike PL, they can measure a pattern's precision on seed questions by checking the correspondence between the extracted answers and the answers given by the seed. As in other work (e.g. Thelen & Riloff 2002), PL uses the fact that it learns patterns for multiple classes at once to improve precision. The particular way we use multiple classes to estimate a pattern's precision (Equation 1) is similar to that of (Lin, Yangarber, & Grishman 2003). A unique feature of our approach is that our heuristic is computed solely by searching the Web for seed values, instead of searching the corpus for each discovered pattern.

A variety of work in information extraction has been performed using more sophisticated structures than the simple patterns that PL produces. W*rapper induction algorithms* (e.g. Kushmerick, Weld, & Doorenbos 1997; Muslea, Minton, & Knoblock 1999) attempt to learn wrappers that exploit the structure of HTML to extract information from Web sites. Also, a variety of *rule-learning* schemes (e.g. Soderland 1999; Califf & Mooney 1999; Ciravegna 2001) have been designed for extracting information from semi-structured and free text. In this paper, we restrict our attention to simple text patterns, as they are the most natural fit for our approach of leveraging Web search engines for both extraction and PMI assessment. For extraction, it may be possible to use a richer set of patterns with Web search engines given the proper query generation strategy (Agichtein & Gravano 2003); this is an item of future work.

Lastly, the work described in this paper is an extension of previous results showing that learning extractors can increase the coverage of the KnowItAll system (Etzioni et. al 2004b). Here, we extend those results by applying PL to learn discriminators, adding a theoretical model for choosing which discriminators to apply, and showing experimentally that pattern learning and the theoretical results have positive impacts on accuracy.

6. Conclusions & Future Work

The addition of pattern learning (PL) improves both the coverage and accuracy of our baseline system. Learned patterns boost coverage by 50% to 80%, and decrease the classification errors of PMI assessment by 28% to 35%. Also, as suggested by theoretical results, the *MU* heuristic reduces classification errors by an additional 19% to 35%, for an overall error reduction of 47% to 53%.

The work presented here offers several directions for future work, most notably generalizing PL to extract patterns for *N*-ary predicates and developing improved methods for automatically estimating the precision and recall of patterns. Also, the success of the *MU* heuristic for discriminator ordering suggests avenues for improving information extraction systems in general. A similar framework could be used to optimize choices between executing extractors, discriminators, or performing pattern learning, depending on the constraints and objectives of the information extraction system.

Acknowledgements

This research was supported in part by NSF grants IIS-0312988 and IIS-0307906, DARPA contract NBCHD030010, ONR grants N00014-02-1-0324 and N00014-02-1-0932, and a gift from Google. Google generously allowed us to issue a large number of queries to their XML API to facilitate our experiments. We thank Jeff Bigham, Tal Shaked, Ian Simon, and Alex Yates for comments on previous drafts.

References

Agichtein, E., and Gravano, S. 2000. Snowball: Extracting relations from large plain-text collections. *Proc. 5th ACM Intl. Conf. on Digital Libraries.*

Agichtein, E., and Gravano, S. 2003. Querying Text Databases for Efficient Information Extraction. *Proc. ICDE-2003.*

Brin, S. 1998. Extracting patterns and relations from the WWW. *Proc. 1998 Intl Wkshp. on the Web and Databases.*

Califf, M. and Mooney, R. 1999. Relational learning of pattern-match rules for information extraction. *Proc. AAAI-99.*

Ciravegna, F. 2001. Adaptive information extraction from text by rule induction and generalisation. *Proc. IJCAI-2001.*

Etzioni, O. 1991. Embedding Decision-Analytic Control in a Learning Architecture. *Artificial Intelligence* 49(1-3): 129-159.

Etzioni, O.; Cafarella, M.; Downey, D.; Kok, S.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.; and Yates, A. 2004a. Web-scale information extraction in KnowItAll. *Proc. WWW-2004*.

Etzioni, O.; Cafarella, M.; Downey, D.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.; and Yates, A. 2004b. Methods for domain-independent information extraction from the Web: An Experimental Comparison. *Proc. AAAI-2004*

Guo, A. 2002. Active Classification with Bounded Resources. *Proc AAAI 2002 Symp. on Information Refinement and Revision for Decision Making.*

Hearst, M. 1992. Automatic acquisition of hyponyms from large text corpora. *Proc 14th Intl. Conf. on Computational Linguistics,* 539-545.

Heckerman, D.; Breese, J. S.; and Rommelse, K. 1994. Troubleshooting under uncertainty. *Technical Report MSR-TR-94-07*, Microsoft Research.

Kushmerick, N.; Weld, D.; and Doorenbos, R. 1997. Wrapper induction for information extraction. *Proc. IJCAI-97,* 729-737.

Lin, W.; Yangarber, R.; and Grishman, R. 2003. Bootstrapped learning of semantic classes. *Proc. ICML-2003 Wkshp on The Continuum from Labeled to Unlabeled Data.*

Muslea, I.; Minton, S.; Knoblock, C. 1999. A Hierarchical Approach to Wrapper Induction. *Proc. 3rd Intl. Conf on Autonomous Agents*.

Papadimitriou, C. and Tsitsiklis, J. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441 – 450.

Ravichandran, D., and Hovy, D. 2002. Learning surface text patterns for a question answering system. *Proc 40th ACL Conf.*

Riloff, E., and Jones, R. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. *Proc. AAAI-99.*

Soderland, S. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning* 34(1-3):233-272.

Thelen, M., and Riloff, E. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. *Proc. 2002 Conf. on Empirical Methods in NLP.*

Turney, P. 2000. Types of cost in inductive concept learning. *Proc. Wkshp. on Cost Sensitive Learning at ICML-2000.*

Turney, P. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. *Proc. 12th European Conf. on Machine Learning.*