The Use of Web-based Statistics to Validate Information Extraction

Stephen Soderland, Oren Etzioni, Tal Shaked, and Daniel S. Weld

Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
U.S.A.
{soderlan,etzioni,shaked,weld}@cs.washington.edu

Abstract

The World Wide Web is a powerful and readily available text corpus that can be used effectively to validate the output of an information extraction system. We present experiments that explore how pointwise mutual information (PMI) from search engine hit counts can be used in an Assessor module that assigns a probability that an extracted fact or relationship is correct, thus boosting precision. We find that thresholding on PMI scores is more effective in creating features for the Assessor than using probability density models. Bootstrapping can be effective in finding both positive and negative seeds to train the Assessor, performing better than hand-tagging a sample of actual extractions.

Introduction

The World Wide Web is a powerful and readily available text corpus that researchers are beginning to exploit for statistical natural language processing. This has already proved effective in testing possible synonyms (Turney 2001), finding semantic orientation of phrases (Turney 2002), and validating question-answer pairs (Magnini *et al.* 2002). More recently, KNOWITALL (Etzioni *et al.* WWW-2004; Etzioni *et al.* AAAI-2004) has shown that search engine hit counts can be used to assign probability of correctness to automatically extracted facts, for a large boost in precision of an information extraction (IE) system.

But in order to use Web-scale statistics as an IE validation tool, a number of questions arise. How can search engine queries be formulated that will provide corroboration for an extracted fact or relation? How can these queries be generated automatically for new domains? How can the hit counts thus obtained be transformed into features for a classifier? Can bootstrapping be used to obtain seeds to train the classifier, in particular obtaining negative seeds?

In this paper, we report on experiments that explore answers to these questions. These experiments use data extracted by the KNOWITALL system, but apply broadly to IE systems in general. KNOWITALL validates extracted instances of a class or relation by computing pointwise mutual information (PMI) from hit counts as defined by (Turney 2001). Let I be an instance and D be a discriminator phrase that is a strong indicator of the target class or relation.

$$\text{PMI}(I, D) = \frac{|\text{Hits}(D + I)|}{|\text{Hits}(I)|} \quad (1)$$

The PMI score is the number of hits for a query that combines the discriminator and instance, divided by the hits for the instance alone. This can be viewed as the probability that $(D + I)$ will be found on a Web page that contains I.

We describe the KNOWITALL system briefly and how its Assessor module uses PMI scores as features for a naive Bayesian probability update. We present a series of experiments to evaluate the following:

- Continuous probability densities as features for the Assessor
- Thresholded PMI scores as features: either single threshold, or one threshold near positive training and one near negative training
- The optimal number of discriminators to use
- Bootstrap training of the Assessor and sensitivity to noise in the training
- Source of negative training: sample from extraction errors, or use positive training from other classes

We found thresholding of PMI scores to be superior to using a continuous probability density. A single threshold gives high precision up to around 0.80 recall, and reclassifying the remaining instances with two thresholds maintains higher precision for the tail of the recall-precision curve. Both recall and precision increase as more discriminators are added, as this helps overcome the bias inherent in individual discriminators.

A combination of generic patterns and bootstrapping are effective in tailoring the validation to specific domains, although manual inspection of the bootstrapped seeds may be needed as noisy seeds can hurt performance. We were surprised to find that selecting negative training from among the seeds of other classes is more effective than using actual "near misses" of the IE system. Overall, we find that Web-based statistics can be highly effective in boosting the precision of an IE system.

```
NP1 "such as" NPList2
    & head(NP1)= plural(Class1)
    & properNoun(head(each(NPList2)))
=> instanceOf(Class1, head(each(NPList2)))
keywords: "plural(Class1) such as"
```

Figure 1: This generic extraction pattern can be instantiated automatically with the (pluralized) class name to create a domain-specific extraction rule. For example, if `Class1` is set to "City" then the rule spawns the search engine query "cities such as", downloads the Web pages, and extracts every proper noun immediately following that phrase as a potential city.

KnowItAll System

KNOWITALL is an autonomous, domain-independent system that extracts facts, concepts, and relationships from the Web. The input to KNOWITALL is a set of classes and relations that constitute its *focus* and a set of generic extraction patterns, some of which were adapted from the hyponym patterns of (Hearst 1992). A bootstrap phase instantiates extraction rules for each class or relation from these generic patterns, and uses them to find seeds to train the Assessor module. KNOWITALL's PatternLearning module (Downey *et al.* 2004) takes extractions from generic rules as seeds, and learns domain-specific patterns that serve as both extraction rules and discriminator phrases. The experiments presented here focus on generic rules and discriminators, and on unary predicates, although KNOWITALL rules can extract N-ary predicates as well.

The *Extractor* module automatically formulates queries based on its extraction rules. Each rule has an associated search query composed of the rule's keywords. For example, if the pattern in Figure 1 were instantiated for the class `City`, it would lead KNOWITALL to 1) issue the search-engine query "cities such as", 2) download in parallel all pages named in the engine's results, and 3) apply the extraction rules to the appropriate sentences on each downloaded page.

The *Assessor* module uses statistics computed by querying search engines to assign a probability to each extracted fact or relation. The Assessor uses a form of *pointwise mutual information* (PMI) between words and phrases that is estimated from Web search engine hit counts in a manner similar to Turney's PMI-IR algorithm (Turney 2001). The Assessor computes the PMI between each extracted instance and multiple *discriminator phrases* associated with the class (such as "city of" for the class `City`). These mutual information statistics are treated as features that are input to a *Naive Bayes Classifier*.

Experiments reported in (Etzioni *et al.* WWW-2004) tested KNOWITALL on five classes: `City`, `USState`, `Country`, `Actor`, and `Film`. Web-based validation was effective in maintaining high precision at high recall levels for these classes. In this paper we present experiments using alternate versions of the Assessor on two of these data sets: `City` with 19,090 instances of which 71% are true positives, and `Country` with 551 instances of which 34% are true positives.

These classes were the most convenient for experiments that evaluate a range of parameter settings and methods, since we could use the Tipster Gazetteer in creating tagged data for evaluation. Only about 70% of the correct city names extracted by KNOWITALL were found in Tipster, so we needed to hand-tag at least a sample of the entries not in Tipster. We were unable to use the Internet Movie Database as an oracle for `Actor` and `Film`, because of the high false negative rate of the code we developed to automatically query IMDB.

We now describe the Assessor in more detail.

Using PMI to Validate Extractions

The Web is a rich source of co-occurrence statistics, computing pointwise mutual information (PMI) from search engine hit counts as in Equation 1. Where Turney used the AltaVista NEAR operator for PMI computation, we found that embedding the instance directly in the discriminator phrase provides stronger evidence. Consider the discriminator phrase "cities such as X", taken from the search query associated with an extraction rule for `City`, and two proposed instances: "Los Angeles", and "California". Each instance has about 12 million hits, but "cities such as Los Angeles" has 4,340 hits, giving a PMI score of 3.6E-4, while "cities such as California" has only 22 hits, giving PMI 1.9E-6. This would cause a system to have much higher belief that Los Angeles is a city than that California is a City.

Discriminator phrases can also be used to validate binary predicates. For example the predicate `Stars-In(Actor, Film)` might have discriminators such as "X in Y" or simply "X Y", where X is replaced by the actor's name and Y by the film. The phrase "Harrison Ford in Star Wars" will have relatively high hit counts, while "Harrison Ford in Jurassic Park" has hardly any hits. The Assessor combines evidence from binary discriminators together with the probability that each argument in a binary predicate is of the proper class.

The PMI-IR algorithm is sufficient to compare the relative likelihoods of two instances, but needs to be extended to compute the probability that a given instance is correct. To do this, we calibrate PMI scores for each discriminator on a training set of known positive and negative instances (seeds). We explore alternate ways to do this in the following section.

KNOWITALL combines evidence from multiple discriminators (and possibly other evidence) in a "naive Bayesian" probability update. Given n observed features $f_1 \ldots f_n$, which are assumed conditionally independent, the Assessor uses the following equation to calculate the expected truth of an atomic formula ϕ, where $P(\phi)$ is the prior probability of the rule producing correct extractions:

$$P(\phi|f_1, f_2, \ldots f_n) = \frac{P(\phi)\prod_i P(f_i|\phi)}{P(\phi)\prod_i P(f_i|\phi) + P(\neg\phi)\prod_i P(f_i|\neg\phi)} \quad (2)$$

To use this equation, we need to transform raw PMI score into two conditional probabilities $P(f_i|\phi)$ and $P(f_i|\neg\phi)$, and train these on a set of positive and negative seeds. These represent the probability of seeing a given PMI score if the instance is valid, and the probability if it is incorrect. We now turn to methods of implementing this.

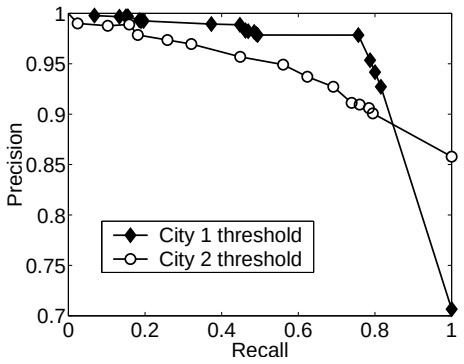

Figure 2: Using a single threshold to turn PMI scores into Bayesian features gives higher precision than two thresholds except for the highest recall levels.

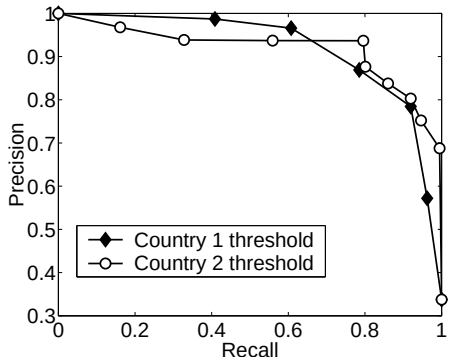

Figure 3: The results from Figure 2 hold for the class `Country` as well.

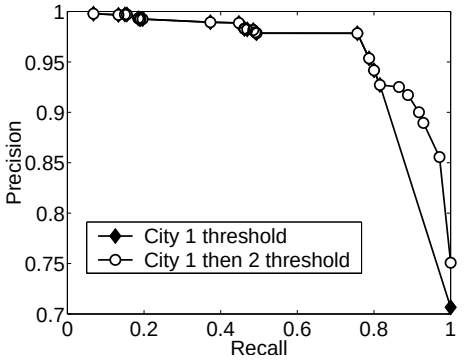

Figure 4: A hybrid method begins with single thresholds, then falls back to double thresholds. This raises precision at high recall while maintaining high precision at low recall.

Creating Features for the Assessor

Our method to turn a PMI score into the conditional probabilities needed for Equation 2 is straightforward. The Assessor takes a set of 10 positive and 10 negative seeds for each class and finds a threshold on PMI scores that splits the positive and negative seeds. It then uses a tuning set of another 10 positive and 10 negative seeds to estimate $P(PMI > thresh|class)$ and $P(PMI > thresh|\neg class)$, by counting the positive and negative seeds (plus a smoothing factor) that are above or below the threshold.

One vs. Two Threshold Method: For many discriminators there is a large gap between the lowest PMI score for positive seeds and the highest PMI for negative seeds (recall the two orders of magnitude difference between "Los Angeles" and "California" for the discriminator "cities such as X"). The Assessor finds two thresholds: *threshA* that maximizes the ratio (p + 1) / (n + 1), where p is the number of positive seeds with PMI $\geq$ threshA and n is the number of negative seeds with PMI $\geq$ threshA. Similarly $threshB$ maximizes the ratio of negative to positive seeds with PMI $\leq$ threshB. KNOWITALL takes the average of threshA and threshB as the threshold for the discriminator.

We noticed a problem with using a single threshold: an instance with PMI just below the threshold and one just above might get radically different probabilities, leading to some misclassifications. We experimented with an alternate method that uses two thresholds. The Assessor estimates the conditional probabilities for PMI $>$ threshA, for PMI $\leq$ threshB, and for PMI between the two thresholds. In cases where there are no training instances between the two thresholds, the conditional probability is the same for $class$ and $\neg class$, adding no information to the probability calculation in Equation 2.

Figure 2 compares the single threshold method with the two threshold method for the class `City`. In each case the Assessor trained a set of 24 generic discriminators that were derived from the class names "city" and "town" to the right or left of the instance (singular and plural) and from the search queries associated with each generic extraction rule derived from those class names. The Assessor retained the five best discriminators to use in classifying new instances. The graph measures precision and recall where the ranking function is the probability assigned by the Assessor.

The single threshold method gives superior precision for instances up to about recall 0.80, but has difficulty distinguishing the positive instances beyond that. The two-threshold method is more effective with the low-PMI instances at the high recall end of the graph. The results are similar for the class `Country` as shown in Figure 3.

We tried a hybrid approach that begins by classifying all instances with a single threshold, then takes the instances that are below threshold for more than half of the discriminator and reclassifies them using two thresholds. Figure 4 shows that this does have the desired effect. With two-threshold reclassification, the curve for `City` is identical to the single threshold curve up to recall 0.81. The two-threshold method is able to further distinguish positive from negative instances, maintaining precision of 0.86 at recall 0.97. The class `Country` has similar results, with precision raised from 0.57 to 0.73 at recall 0.96 and from 0.34 to 0.67 at recall 1.0.

Continuous Probability Density Method: Modeling the conditional probabilities as continuous probability density functions seems more attractive than probabilities that jump

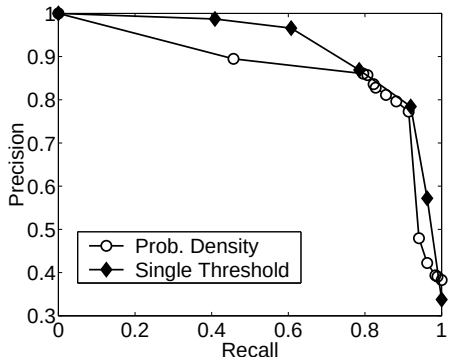

Figure 5: Using a continuous probability density function causes a drop in precision at both ends of the recall-precision curve.

abruptly at PMI thresholds. Fitting such a model to limited training, on the order of 20 positive and 20 negative seeds, proves difficult, however. We computed a probability density function (PDF) by smoothing a Gaussian kernel with bandwidth computed using a rule of thumb suggested by (Turlach 1993)[1]. We truncated the PDF for values less than 0 and rescaled the remaining area such that it would integrate to 1. Similar PDFs were done for the negative training.

$$PDF_{f_i,\phi}(x) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-x_j)^2}{2\sigma^2}} \quad (3)$$

The results of using this continuous PDF equation on the class `Country` is shown in Figure 5, with a comparison to the single-threshold method. Each used the same five discriminators trained on the same seeds. Similar results for `City` were reported in (Etzioni *et al.* WWW-2004).

The point at which the positive and negative PDFs for a given discriminator intersect (midpoint) is analogous to the thresholds described earlier. As PMI scores move farther from this midpoint, the ratio between PDFs change, giving different levels of confidence.

Although in principle, accounting for the magnitude of the PMI score is advantageous, we found a drop in precision at both ends of the recall-precision curve. In practice the PDF for the negative training points is particularly non-representative of the underlying distribution and results in extreme and unpredictable ratios for PMI scores too far from the midpoint. One PMI score that is an outlier can dominate the overall classification which results in the clumps we see, as well as some of the errors.

Synergy of Multiple Discriminators

Another decision in Web-scale validation is the number of discriminators to use. This is partly an efficiency issue: validating an extraction using k discriminators requires $k+1$ hit count queries, one for the instance itself and one each for the discriminators; each additional discriminator means an extra query to a search engine. Processing time in KNOWITALL is dominated by the time taken for validation queries. There is another issue as well that involves synergy between discriminators.

Discriminators that use a variety of words associated with a class or relation can help overcome a problem of polysemy. The term "country" might refer to countryside or to a music genre. We used discriminators with the plural form "countries" as well as an alternate class name "nation" and "nations". Instances that fool one discriminator phrase are not likely to fool all discriminators.

Another artifact of discriminators is the difference between *left-handed* discriminators that form a prefix of the instance and *right-handed* discriminators that form a suffix. The incorrect instance of `City` "Los" will have high PMI for a left-handed discriminator ("city Los"), but low PMI for a right-handed discriminator ("Los city"). The fragment "Angeles" will fool right-handed discriminators, but not left-handed. The Assessor keeps a balance between right-handed and left-handed discriminators when automatically selecting the k best discriminators.

We ran an experiment where the number of discriminators was varied from 1 to 24. Figure 6 shows the results for `Country` for 1, 3, 5, and 10 discriminators. Performance increases only slightly after 5 discriminators. (Downey *et al.* 2004) found good results from dynamically deciding how many discriminators to use for each extraction, favoring those with greatest uncertainty.

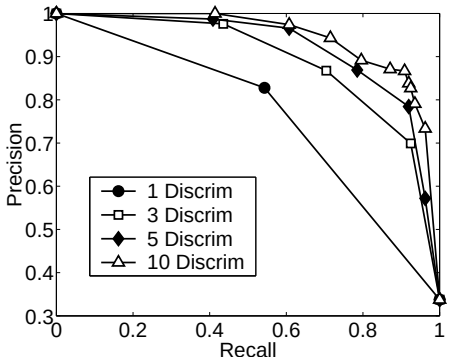

Figure 6: Adding more discriminators raises both recall and precision, with smaller increases after 5 discriminators.

Bootstrapping and Noise Tolerance

Another important issue is robustness and noise tolerance, particularly when bootstrapping is used to select training for the Assessor. KNOWITALL begins with a bootstrap step that instantiates a set of extraction rules and a set of discriminators from generic patterns. To select seeds for the Assessor, KNOWITALL extracts a batch of unverified instances using the generic rules and computes the PMI for each of the generic discriminators. This produces a set of instances ranked by average PMI over all the discriminators.

KNOWITALL selects n seeds from the top m instances, then trains all discriminators on those seeds and selects the best k discriminators (we used $k=5$, $n=20$, and $m=60$).

[1] $h = 1.06(min(\hat{\sigma}, \hat{R}/1.34))N^{-1/5}$ where $\hat{\sigma}^2$ and $\hat{R}$ are the variance and interquartile range over the $N=20$ positive training points $\{x_j\}$

This process can be iterated by reranking the instances by the probability computed from those k discriminators and picking a new set of seeds to retrain all discriminators. We discuss the source of negative seeds later in this section.

Figure 7 compares performance for `Country` trained on three different sets of seeds. The first pass of bootstrapping had 30% noise: 6 errors out of 20 seeds ("EU", "Middle East Countries", "Iroquois", and other instances semantically related to nation or country). A second bootstrap pass produced seeds with 10% noise. Manual review to reject incorrect seeds produced the set with no noise. The set of seeds with 10% noise produced precision comparable to no noise up to about recall 0.75, then degraded badly. A single bootstrapping pass produced seeds with no errors for the classes `City`, `USState`, and `Actor`, however. More research is needed to find a bootstrapping method that eliminates noise across all classes.

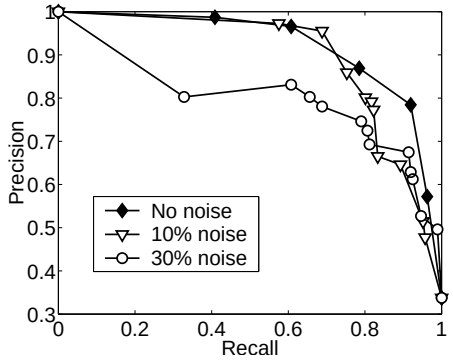

Figure 7: The Assessor can tolerate 10% noise in bootstrapped training seeds up to recall 0.75, but performance degrades after that.

Another question that troubled us is the source of negative seeds. No automatic means of producing useful "near misses" comes to mind. Our solution was to train the Assessor on multiple classes at once; KNOWITALL finds negative seeds for a class by sampling positive seeds from other classes, as in (Lin *et al.* 2003). We take care that each class has at least one semantically related class to provide near misses. In these experiments, `Country` gets negative seeds from `City`, `USState`, `Actor`, and `Film`, and so forth.

We tried the following alternative method of finding negative seeds. KNOWITALL runs its Extractor module to produce a set of unverified instances, then takes a random sample of those instances, which are hand-tagged as seeds. This training set has the added advantage of a representative proportion of positive and negative instances.

Figure 8 shows an experiment where a random sample of 40 extractions were hand-tagged as seeds. These seeds were then removed from the test set for that run. Surprisingly, the recall-precision curve is somewhat worse than selecting negative seeds from the other classes.

Related Work

KNOWITALL builds on work of (Turney 2001, 2002, 2003) whose PMI-IR uses search engine hit counts to compute

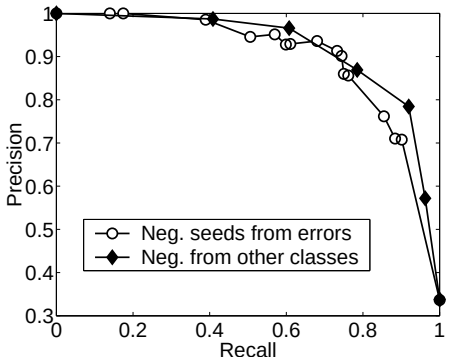

Figure 8: Using negative seeds that are taken from seeds of other classes works better than tagging actual extraction errors as negative seeds.

pointwise mutual information that measures the degree of *non-independence* between a pair of words. Turney used PMI from hit counts to select among candidate synonyms of a word, and to detect the semantic orientation of a phrase by comparing its PMI with positive words (*e.g.* "excellent") and with negative words (*e.g.* "poor").

Other researchers have recently made use of PMI from hit counts. (Magnini *et al.* 2002) validate proposed question-answer pairs for a QA system by learning "validation patterns " that look for the contexts in which the proposed question and answer occur in proximity. Like Turney, they use the AltaVista NEAR operator to allow a limited amount of arbitrary text between terms in their queries.

(Uryupina 2003) classifies proposed instances of geographical classes by embedding the instance in discriminator phrases, much as KNOWITALL does. Both the raw hit counts and normalized counts (PMI scores) are then given as features to the Ripper classifier. Uryupina's system includes a bootstrapping cycle that begins with 100 hand-tagged seeds, then alternately learns patterns to find and validate more instances, and learns new patterns from those instances.

KNOWITALL is also related to a growing body of research on bootstrapping methods for natural language processing. (Jones *et al.* 2003) gives a good overview of the choices of methods in bootstrap learning. Bootstrap is an iterative approach that alternates between learning rules from a set of instances, and finding instances from a set of rules (Riloff and Jones 1999; Agichtien and Gravano 2000; Brin 1998). Where previous bootstrapping methods begin with a small set of hand-tagged seeds, KNOWITALL begins with a small set of domain-independent extraction patterns and uses that to find its initial set of seeds.

Discussion and Open Questions

We have found that Web-based pointwise mutual information (PMI) statistics can be effective in validating information extraction (IE). This paper explores alternate choices in implementing PMI in an IE system.

The lessons learned were:

- Thresholding of PMI scores produce better features for the Assessor than a continuous probability density model.
- A single threshold gives better precision than setting two thresholds (a lower bound on positive seeds and an upper bound on negative seeds). This holds on all but the tail of the recall-precision curve, where two thresholds give higher precision.
- Both recall and precision improve as the number of discriminators increases, although there are smaller increases after 5 discriminators.
- Bootstrap training can be effective in finding seeds to train the Assessor, although manual inspection of the seeds is important. The Assessor is sensitive to noise in its training.
- Selecting negative training from positive seeds of other classes gives better classification than hand-tagging extraction errors as negative seeds.

Several open questions remain about the use of PMI for IE systems. Even with the entire Web as a text corpus, the problem of sparse data remains. The most precise discriminators tend to have low PMI scores for positive instances, often as low as 10^{-5} or 10^{-6}. This is not a problem for prominent instances that have several million hits on the Web. If an instance is found on only a few thousand Web pages, the expected number of hits for a positive instance will be less than 1 for such a discriminator. This leads to false negatives for the more obscure positive instances.

Several levels of backing off are possible to help overcome this. If a discriminator produces 0 hits, a more general discriminator can be tried. If the discriminator "X is a city" with threshold 2.2E-5 is too specific for a low frequency instance, more general discriminators such as "X city" with threshold 9.5E-3 may be informative. If further backing off is needed, "X NEAR city" is more general, and sending X and "city" as separate query terms is the most general discriminator query.

A different problem with using PMI is homonyms — words that have the same spelling, but different meanings. For example, Georgia refers to both a state and country, Normal refers to a city in Illinois and a socially acceptable condition, and Amazon is both a rain forest and an on-line shopping destination. When a homonym is used more frequently in a sense distinct from the one we are interested in, then the PMI scores may be low and may fall below threshold. This is because PMI scores measure whether membership in the class is the *most common* meaning of a noun denoting an instance, not whether membership in the class is a *legitimate but less frequent* usage of that noun.

Another issue is in the choice of a Naive Bayes Classifier, which has flexibility in combining varied evidence, but generates probabilities that tend towards 0.0 and 1.0 since the independence assumption about features is usually unrealistic. We are considering using SVD techniques to reduce the dimensionality of the feature space to compensate for non-independent features, and a nearest-neighbor recalibration of the probability estimates.

Acknowledgments

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.

References

E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

S. Brin. Extracting patterns and relations from the World-Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases*, 1998.

D. Downey, O. Etzioni, S. Soderland, and D. Weld. Learning text patterns for Web information extraction and assessment. *To appear in AAAI-2004 Workshop on Adaptive Text Extraction and Mining*, 2004.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in KnowItAll (preliminary results). *To appear in the 13th Intl. World Wide Web Conference*, 2004.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Methods for domain-independent information extraction from the Web: an experimental comparison. *To appear in the 19th National Conference on Artificial Intelligence*, 2004.

M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Intl. Conf. on Computational Linguistics*, 539–545, 1992.

R. Jones, R. Ghani, T. Mitchell, E. Riloff. Active learning for information extraction with multiple view feature sets. In *ECML-03 Workshop on Adaptive Text Extraction and Mining*, 2003.

W. Lin, R. Yangarber, and R. Grishman. Bootstrapped learning of semantic classes. In *Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data*, 2003.

B. Magnini, M. Negri, and H. Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 425-432, 2002.

E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

B.A. Turlach. Bandwidth selection in kernel density estimation: A review. Discussion Paper 9317, Institut de Statistique, Voie du Roman Pays 34, B-1348 Louvain-la-Neuve. 1993.

P. D. Turney. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning*, 2001.

P. D. Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Assoc. for Computational Linguistics*, 417-424, 2002.

P. D. Turney and M. Littman. Measuring praise and criticism: Inference of semantic orientation from association. In *ACM Transactions on Information Systems (TOIS)*, 21(4), 315–346, 2003.

O. Uryupina. Semi-supervised learning of geographical references within text. In *NAACL-03 Workshop on the Analysis of Geographic References*, 2003.