Improving Automatic Query Classification via Semi-supervised Learning

Steven M. Beitzel
Eric C. Jensen, Ophir Frieder
Information Retrieval Laboratory
Computer Science Department
{steve,ej,ophir}@ir.iit.edu

David D. Lewis
Abdur Chowdhury, Aleksander Kołcz
America Online, Inc.
davelewis@daviddlewis.com
{cabdur,arkolcz}@aol.com

Abstract

Accurate topical classification of user queries allows for increased effectiveness and efficiency in general-purpose web search systems. Such classification becomes critical if the system is to return results not just from a general web collection but from topic-specific back-end databases as well. Maintaining sufficient classification recall is very difficult as web queries are typically short, yielding few features per query. This feature sparseness coupled with the high query volumes typical for a large-scale search service makes manual and supervised learning approaches alone insufficient. We use an application of computational linguistics to develop an approach for mining the vast amount of unlabeled data in web query logs to improve automatic topical web query classification. We show that our approach in combination with manual matching and supervised learning allows us to classify a substantially larger proportion of queries than any single technique. We examine the performance of each approach on a real web query stream and show that our combined method accurately classifies 46% of queries, outperforming the recall of best single approach by nearly 20%, with a 7% improvement in overall effectiveness.

1. Introduction

Understanding the topical sense of user queries is a problem at the heart of web search. Successfully mapping incoming general user queries to topical categories, particularly those for which the search engine has domain-specific knowledge, can bring improvements in both the efficiency and the effectiveness of general web search. Many existing web search engines must automatically route each incoming query to an appropriate set of topic-specific back-end databases and return a merged listing of results. Correct routing decisions result in reduced computational and financial costs for the search service since it is impractical to send every query to every backend-database in the (possibly large) set. Accurate topical classification of a query also has broad potential for improving the effectiveness of the result set, as results from topic-specific databases can be given preference in presentation.

With the goal of automatically classifying a meaningful portion of the web query stream, techniques based solely on labeled data such as manual classification and supervised learning are too limited to be of much practical use, and vastly more data are required to make a significant impact on the problem. Our key contribution, introduced as an extended abstract in [1], is the use a very large web search engine query log as a source of unlabeled data to aid in automatic classification. Large web query logs are a gold mine of potential information that is hard to utilize with simple techniques alone. We develop a rule-based automatic classifier produced using selectional preferences mined from the linguistic analysis of an unlabeled query log containing hundreds of millions of queries. This technique is used in combination with exact matching against a large set of manually classified queries and a weighted automatic classifier trained using supervised learning to achieve the most effective classification possible. Our combined approach outperforms each individual method and achieves a high level of recall. This mitigates the recall problem due to feature-sparseness encountered in prior query classification studies, and allows us to effectively classify a much larger portion of general web queries.

2. Prior Work

In the case of large-scale web search, building a classification system that can automatically classify a large portion of the general query stream with a reasonable degree of accuracy is particularly challenging. The web's content and user-base are constantly changing [2], web traffic at major search engines often reaches hundreds of millions of queries per day [3], and web queries are topically ambiguous and very short [4]. This complicates manual categorization of queries (for study, or to provide training data), and poses great challenges for automated query classification [5, 6].

In Table 1 we provide a topical breakdown based on a manually classified sample from one week's worth of queries from the AOL web search service, which has a large number of backend databases with topic-specific information. Accurate classification of queries into these categories (as well as more specific categories further down the hierarchy) would improve effectiveness of search results, reduce the computational and financial costs of system operation, and provide new opportunities for revenue.

Table 1: Query Stream Breakdown

Autos	3.46%	Personal Fin.	1.63%
Business	6.07%	Places	6.13%
Computing	5.38%	Porn	7.19%
Entertainment	12.60%	Research	6.77%
Games	2.38%	Shopping	10.21%
Health	5.99%	Sports	3.30%
Holidays	1.63%	Travel	3.09%
Home&Garden	3.82%	URL	6.78%
News&Society	5.85%	Misspellings	6.53%
Orgs.&Insts.	4.46%	Other	15.69%

There have been a number of studies involving topical query classification, but a common theme to all of them is their reliance on some form of labeled data for experimentation, often in the form of result documents for a query, clickthrough data from sessions, or manual, human-assessed classification efforts.

Gravano, et al. used machine learning techniques in an automatic classifier to categorize queries by geographical locality [5]. They sampled small training, tuning, and testing sets of a few hundred queries each from a 2.4 million-query Excite™ log from December of 1999 and mined the result documents for classification features. Their analysis revealed that data sparseness was a problem for their classifier, noting in particular that most queries were very short, and, specific to their geographical task, queries rarely contain key trigger words that are likely to identify whether or not a query is "local" or "global" in geographical scope. They concluded by positing that for such an automatic classification system to be successful, it would have to draw on auxiliary sources of information, such as user feedback or supplemental databases, to compensate for the small number of available features in a single query. This study does make use of a query log, but only a tiny fraction of its data is used, and several result documents from each query are required for the technique to be effective. Techniques of this kind are clearly insufficient for the task of general topical classification.

An alternative to classifying queries into manually defined categories is to allow classes to emerge from a query set itself, through clustering or other unsupervised learning methods. There is a long history of such approaches in Information Retrieval [7], with attempts in the past often stymied by the smaller query logs available. In clustering web queries, the problem is no longer lack of queries, but lack of features in any individual query. This is arguably an even greater problem for unsupervised than supervised approaches, which can at least latch on to individual features correlated with a class.

Some query clustering methods have attacked this problem by clustering "session data", containing multiple queries and click-through information from a single user interaction. Beeferman and Burger [8] mined a log of 500,000 click-through records from the Lycos™ search engine and used a bipartite-graph algorithm to discover latent relationships between queries based on common click-through documents linking them together. Unfortunately, only anecdotal results were reported in the study. A similar approach was used by Wen, et al. [9-11], who also took into account terms from result documents that a set of queries has in common. They concluded that the use of query keywords together with session data is the most effective method of performing query clustering. However, their test collection was an encyclopedia, so the applicability of their results to general web search is limited. Studies of this nature are also limited in their application, as large quantities of continually refreshed session data and result documents, not to mention computationally expensive analysis, would be required to classify each query. For a search service with a daily volume in the hundreds of millions, these approaches are computationally infeasible.

3. Classification Approaches

Prior efforts in classifying general web queries have included both manual and automatic techniques. We now describe the manual and automatic classification approaches used in our experiments. We also introduce a new rule-based automatic classification technique based on an application of computational linguistics for identifying selectional preferences by mining a very large unlabeled query log. We demonstrate that a combination of these approaches allows us to develop an automatic web query classification system that covers a large portion of the query stream with a reasonable degree of precision.

3.1 Exact Matching

The simplest approach to query classification is looking the query up in a database of manually classified queries. At any given time, certain queries (the reader can anticipate some of these) are much more popular than others. By combining manual classification of these queries with acquiring large databases of proper nouns in certain categories (personal names, products, geographic locations, etc.), non-trivial coverage of the query stream can be achieved.

We explored this technique by using 18 lists of categorized queries produced in the above fashion by a team of editors at AOL. We examined the coverage of these categories and found that even after considerable time and development effort, they only represented ~12% of the general query stream. In addition to poor coverage, a key weakness in this approach is that the query stream is in a constant state of flux [2]. Any manual classifications based on popular elements in a constantly changing query stream will become ineffective over time. Additionally, manual classification is very expensive. It is infeasible to label enough queries, and to repeat this process often enough, to power an exact match classification system that covers a sufficient amount of the query stream. Many queries are ambiguous, making assessor disagreements inevitable. Bulk loading of large lists of entities contributes to the problem, through mismatches in category definitions and introduction of possible but low frequency interpretations for queries. This approach is insufficient if the end goal is to develop a high-recall classification system for general web queries.

3.2 Supervised Machine Learning

A natural next approach is to leverage the labeled queries from the exact match system through supervised learning. The idea is to train a classifier on the manual classifications with the goal of uncovering features that enable novel queries to be classified with respect to the categories. A challenge for this approach is that web queries are short, averaging between 2 and 3 terms per query. This leaves a learner with very few features per example.

For our experiments we use the Perceptron with Margins algorithm [12], shown to be competitive with state-of-the-art algorithms such as support vector machines in text categorization and computationally efficient. Our implementation normalizes the document feature vectors to unit length (Euclidean norm), which we have found to increase classification accuracy.

To examine the effectiveness of the supervised learning approach we trained a perceptron for each category from the exact match system, using all queries in a given category as positive examples and all queries not in that category as negative examples.

To realistically evaluate this approach, we developed a test collection that is representative of the general query stream. To determine the number of queries required to achieve a representative sample of our query stream, we calculated the necessary sample size in queries according to the sample size formula given in [13]. By setting our confidence level to 99% and error rate to 1%, we require a sample of at least 16,638 queries.

To build the test set we had a team of human editors perform a manual classification of 20,000 queries randomly sampled from general query stream. These 20,000 queries were then used for testing our perceptron learner trained on the database from the exact match system. (We use this test collection for all experiments in this paper. See section 4.1 Data Sets for a detailed description of this test collection.)

In Figure 1, we show the DET curves (probability of false positive on x-axis, probability of false negative on y-axis) for five categories on the test set. While the perceptron classifier substantially generalized beyond the coverage of the exact match system, we see it can achieve high coverage only at the cost of large numbers of false positives. For the best of the categories shown, Porn, covering 90% of the positive examples leads to classifying around half of non-Porn queries as Porn.

In Figure 3 (in Section 5. Results & Analysis), we illustrate the problem even more graphically. For the category shown (Home & Garden) pushing attempting to cover more than 20% of positive examples leads to a catastrophic drop-off in precision, as the classifier is required to accept queries which match no good features.

The fundamental problem is that the databases of the exact match system, partly by design and partly due to ongoing changes in the query stream, do not systematically cover the space of queries. Important predictor features are inevitably missing or underrepresented. Robust coverage requires an approach that can leverage both the labeled data that is available, and the vast amounts of unlabeled data present in query logs. We examine such an approach in the next section.

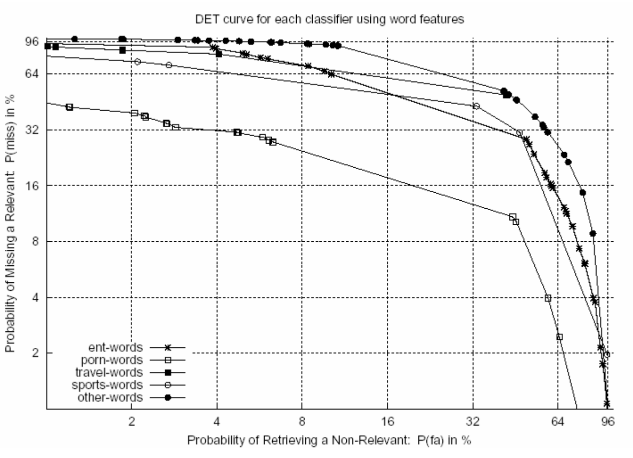

Figure 1: DET Curve for Perceptron Learner on 5 Example Categories

3.3 Selectional Preferences

Document classification typically relies on the weighted combination of evidence from many words. Queries have fewer words and a simpler structure than documents. That suggested we might profitably view classifying a query as more like interpreting the meaning of a linguistic structure than like classifying a document.

Computational linguists have studied many approaches to assigning meanings to linguistic entities. The technique we explored is based on selectional preferences [14]: the tendency for words to prefer that their syntactic arguments belong to particular semantic classes. For instance, the verb eat requires (except in metaphorical uses) that its direct object be something edible. Many techniques for learning selectional preferences from data are computationally efficient, use data in a form that can naturally be derived from query logs and, of particular interest, can make use of unlabeled data.

We describe below how selectional preferences are typically approached in computational linguistics, and how we adapted this technique to query classification.

3.3.1 Selectional Preferences in Computational Linguistics

A selectional preference study begins by parsing a corpus to produce pairs, (x, y), of words that occur in particular syntactic relationships. The verb-object relationship is most widely studied, but others such as adjective-noun have been used [15]. The interest is in finding how x, the context, constrains the meaning of y, its argument.

Resnik [16] presented an influential information theoretic approach to selectional preference. He defines the *selectional preference strength S(x)* of a word *x*, as Equation 1, where *u* ranges over a set *U* of semantic classes. *P(U)* is the probability distribution of semantic classes taken on by words in a particular syntactic position, while *P(U|x)* is that distributions for cases where a contextual syntactic position is occupied by *x*. S(x) is simply the KL divergence [17] of *P(U|x)* from *P(U)*.

$$\begin{aligned} S(x) &= D(P(U|x) \,||\, P(U)) \\ &= \sum_u P(u|x)\log_2\left(\frac{P(u|x)}{P(u)}\right) \end{aligned}$$

Equation 1: Selectional Preference Strength

The ideal approach to estimating the necessary probabilities would be to use a set of *(x,y)* pairs where each *y* has been tagged with its semantic class, perhaps produced by parsing a semantically tagged corpus. The maximum likelihood estimates (MLEs) are:

$$\hat{P}(u) = \frac{n_u}{N} \qquad \hat{P}(u \mid x) = \frac{n_{xu}/N}{n_x/N} = \frac{n_{xu}}{n_x}$$

where *N* is the total number of queries, n_u is number of queries with a word of class *u* in the syntactic position of interest, n_x is the number of queries with *x* providing context for that position, and n_{xu} is the number of queries where both are true. Resnik ([16], Appendix A) discusses other estimates but says they seem to give similar results to the MLE.

Large semantically tagged corpora are rare, however. A more common approach is to use unlabeled data, plus a thesaurus specifying which semantic classes each y can belong to. If a given y has only one class u it can belong to, we can replace each pair (x,y) with the pair (x,u). Some y's will not appear in the thesaurus, and the corresponding (x,y)'s are usually discarded. Conversely, some y's will be ambiguous (belong to multiple semantic classes) and so must be handled specially. Resnik treats each occurrence of such a y as contributing fractionally to the count of each of its word senses, so that:

$$n_u = \sum_{y \in W_u} \frac{n_y}{|U_y|} \qquad n_{xu} = \sum_{y \in W_u} \frac{n_{xy}}{|U_y|}$$

where W_u is the set of words which have *u* as one of their classes, and U_y is the set of classes to which word *y* belongs. So, for instance, if a word *y* belongs to two classes, an occurrence of the word contributes a count of 1/2 to each of the two classes.

3.3.2 Selectional Preference in Query Logs.

Selectional preferences can be used for disambiguation and semantic interpretation. If *x* strongly favors *y*'s that belong to class *u*, then *u* is a good prediction for the class of an ambiguous, or previously unknown, *y* in that context. Indeed, many studies have evaluated the quality of learned selectional preferences by measuring the accuracy with which they can be used for disambiguation [18].

To take advantage of this disambiguation effect to classify queries, we do the following:

1. Convert queries in the log to a set of head-tail (x,y) pairs.
2. Convert the (x,y) pairs to weighted (x,u) pairs, discarding y's for which we have no semantic information
3. Mine the (x,u) pairs to find lexemes that prefer to be followed or preceded by lexemes in certain categories (preferences)
4. Score each preference using Resnik's Selectional Preference Strength and keep the strongest ones

Step 1 is straightforward for queries, *vw*, of length 2. We have only two possible "syntactic" relationships: the first token providing context for the second, or the second providing context for the first. These produce the *forward* pair *(_v,w)* and the *backward* pair *(w_,v),* with the underscore indicates matching at the front or the back of the query, respectively. We keep pairs of the two types separate, and call selectional preferences mined from *(v,w)* pairs *forward preferences*, and those from *(w,v)* pairs *backward preferences.*

If all two token queries were simple noun phrases (a modifier followed by a noun), then forward preferences would capture the degree to which a particular modifier (noun or adjective) constrained the semantics of the head noun. Backward preferences would capture the reverse. In practice, two token queries can arise from a variety of other syntactic sources: verb-object pairs, single words spelled with two tokens, non-compositional compounds, proper names, etc. A user may also intend the query as a pairs of single words in no particular syntactic relation. Typographical errors and other anomalies are also possible. Thus our forward and backward relations inevitably have a murky interpretation.

Longer queries have more structural possibilities. Rather than attempting to parse them, we derived from a query *abc...uvw* all pairs corresponding to binary segmentations, i.e.:

(a, bc...w), (ab, c...vw), ... (abc...v, w)

and

(bc...w, a), (c...w, ab), ... (w, abc...v).

For any given query, most of these pairs get screened out in Step 2.

In Step 2, we replace each pair *(x,y)* with one or more pairs *(x,u)*, where *u* is a thesaurus class. Pairs where *y* is not present in the thesaurus are discarded, and pairs where *y* is ambiguous yield multiple fractionally weighted pairs as discussed in the previous section. Our "thesaurus" is simply our database of manually classified queries, with each query interpreted as a single (often multi-token) lexical item. The possible semantic classes for a lexical item are simply the set of categories it appears under as a query.

In Step 3, we compute *S(x)* for each *x*, as well as the MLE of *P(u|x)* for each *(x,u)* pair seen in the data. We then screen out pairs where $S(x) < 0.5$, a relatively low threshold on selectional preference strength determined by initial tests on our validation set. From each remaining pair we form a rule $[x \rightarrow u{:}P(u|x)]$, which is interpreted as saying that a query matching *x* gets a minimum score of *P(u|x)* for category *u*. If *(x,u)* was a forward pair (i.e. x is *"_v"*), we require *x* to match a prefix of the query for the rule to apply, while if *(x,u)* is a backward pair we require *x* to match a suffix of the rule to apply.

Finally, in Step 4 we use selectional preferences to classify test queries. We attempt to match each forward selectional preference against the initial tokens of the query, and each backward preference against the final tokens of the query. We give the query a score for each category *u* corresponding to the maximum *P(u|x)* value of any rule that matches it. We then compare the maximum *P(u|x)* values for a query against a threshold tuned to optimize classification effectiveness (Section 3.4), and assign the query to all *u*'s with values that exceed the threshold. Tuning is necessary since the *P(u|x)* values are estimates of the probability that a subpart of the query would be viewed as belonging to a category (with categories considered as mutually exclusive), not estimates of the probability that the whole query would be viewed as belonging to that category (with categories considered as overlapping).

The above approach can be compared with conventional rule learning approaches in machine learning [19]. Like rule learning approaches it uses labeled data to learn logical relationships (in our case very simple ones) between predictor features and classes, and like some of these approaches uses weighted rules with conflict resolution by taking maximum score. Our linguistic criterion for filtering rules, *S(x)*, is somewhat different from those used for feature and rule selection in a typical rule learner, particularly since it implies a different view of category structure than the classifier itself uses. Our use of ambiguously labeled and structured training data is very atypical in rule learning. Finally, most rule learners incorporate an inductive bias toward producing small sets of rules, while the short length of queries requires that as many rules (of sufficient quality) be produced as possible. With respect to this last point, our approach has similarities to association rule learning, which emphasizes producing all rules of sufficient quality. The focus in association rule learning is usually not on predictive accuracy, nor are issues of ambiguity typically dealt with.

3.4 Tuning and Combining Classifiers

One ideally tunes a text classifier to the need of particular application [20]. Our exact match classifier either finds a query in its database or doesn't, so no tuning is possible. Our perceptron and SP classifiers, on the other hand, produce a score for a query on each category, and these scores must be converted to classification decisions.[1] Therefore, for each experimental run we set the thresholds of these two classifiers to optimize the particular micro-averaged F_β measure (Sec. 4.2) to be measures. Tuning was done on a sample (Sec. 4.1) distinct from the test set. A single threshold was used for all categories on each classifier. This helped avoid choosing extreme thresholds for categories with little training data, but may have hurt effectiveness to the degree that scores on different categories were not comparable.

Simply tuning the threshold of a classifier cannot make it do well on queries whose important features were not present in the training data. We found that the positive example overlap between each classifier was generally low, suggesting that a combination might lead to greater effectiveness. We tested a simple combined classifier that assigned a query to each category that any of the above three classifiers predicted. The component classifiers were individually tuned to optimize micro-averaged F_β, but no tuning of the combined classifier as a whole was done. More sophisticated approaches are certainly possible.

4. Evaluation

Evaluation in the context of an evolving real-world system is always a challenge. In this section we discuss the choices we made in structuring controlled experiments with our data sets. We give an overview of the datasets and effectiveness measures that we used to evaluate each individual approach, and our combined approach.

4.1 Data Sets

We used two primary data sets for our experiments. The first data set is composed of several hundred thousand queries that were manually classified by human editors into the set of 20 categories listed in Table 1, excepting "URLs" and "Misspellings", as they are not likely to be classified with any accuracy by any automatic method. The queries in the 18 remaining categories provide exact-lookup matches for manual classification, are used as training data for the perceptron, and they define class memberships for lexeme's when mining a query log for selectional preferences.

The second data set is a random sample of 20,000 queries out of the entire query stream for one week. These queries were manually classified into the same set of 18 categories by a team of human editors, and they are used for tuning and evaluating each individual approach. As with our training data set, "URLs" and "Misspellings" were removed from consideration for these experiments. A tuning set was formed from approximately 25% of the remaining queries. This set is used to tune the perceptron and selectional preference classifiers to optimal thresholds. The remaining 75% of queries in this data set were used strictly for evaluation.

4.2 Effectiveness Measures

Overlap between our categories is low, but a query can belong to more than one category. We therefore treat each category as a separate binary classification task. We summarize a classifier's effectiveness on a category k by the number of true positives (TP_k), false positives (FP_k), false negatives (FN_k), and true negatives (TN_k), where $TP_k + FP_k + FN_k + TN_k$ equals the number of test documents. Using these counts, we can define the usual measures recall, $R_k = TP_k / (TP_k + FN_k)$, precision, $P_k = TP_k / (TP_k + FP_k)$, and the F-measure [21]:

$$F_{\beta,k} = \frac{(\beta^2+1)\times TP_k}{(\beta^2+1)\times TP_k + FP_k + \beta^2 FN_k}$$

Equation 2: F-Measure

where β lets us specify the relative importance of recall and precision. Values of β lower than 1.0 indicate precision is more important than recall, while values of β greater than 1.0 indicate the reverse.

To summarize effectiveness across all categories we use the micro-averaged [22] values of recall, precision, and F over the set of categories. Micro-averaging and macro-averaging give similar results on our data. Note that the micro-averaged value of F_β is not derived from the micro-averaged precision and recall values.

5. Results & Analysis

Table 2 shows the effectiveness, as measured by micro-averaged F_1, of the three individual classification approaches and our disjunctive combination scheme. All three tunable methods (perceptron learner, selectional preferences, and our disjunctive combined approach) were tuned to optimize micro-averaged F_1

[1] The perceptron algorithm itself sets a threshold on each linear model, but these thresholds implicitly minimize error rate, not F-measure.

on the validation set. To provide additional insight we also show micro-averaged recall and micro-averaged precision, though it is important to remember that averaged F_1 values are not simple functions of averaged recall and precision values.

Table 2: Classification Effectiveness for Each Technique

	Micro F1	Micro Precision	Micro Recall
Exact Match	.0749	.3079	.099
Perceptron	.1135	.1913	.279
SP	.1093	.1524	.3856
Combined	.1215	.1651	.4602
Over Best	7.05%	-46.38%	19.35%
Over Worst	62.22%	8.33%	364.85%
Over Mean	22.44%	-23.99%	80.80%

These results show that the combined approach outperforms all three individual techniques, and in particular it achieves approximately 20% higher recall than any single approach. This large increase in recall suggests that the combined approach may in fact be a tenable solution to the recall problem that has hindered past efforts at query classification. In order to examine the performance of each approach when precision and recall are not equally weighted, we must analyze each method's performance for different values of β.

As discussed, we can specify a different tradeoff between recall and precision by varying the value of β in the F-measure, and retuning the automatic classification approaches and the combined approach to optimize that measure on the validation set.

Figure 2 plots values of β against the test set micro-averaged Fβ value of a classifier tuned to optimize micro-averaged Fβ on the validation set. We can see that the combined approach increasingly dominates the perceptron approach as higher levels of recall are desired (higher β). This makes sense, since the disjunctive combination achieves recall at least as good (and almost certainly better) than that of its best component, but precision no better (and almost certainly worse) than that of its best component. The combination also maintains a consistent edge over the selectional preferences for all values of β, and keeps pace with the machine learning approach when precision is more important (lower β).

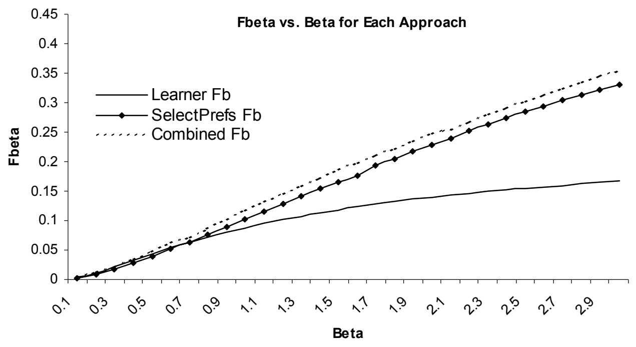

Figure 2: F_β vs. β for Each Classification Technique

Focusing on individual categories gives more insight into the properties of the various approaches. Figure 3 plots test set recall and precision for the perceptron and selectional preference classifiers on the Home category. We see that the two approaches are remarkably complementary. The perceptron classifier (as trained on the database of the exact match system) can achieve high precision at low recall levels, but as discussed in Section 3.2 fails catastrophically if high recall is demanded. The selectional preference classifier never achieves very high precision, but can maintain some precision out to high recall levels. Additionally, selectional preference classifiers cannot make classification decisions on single-term queries. If evaluated over multi-term queries alone, higher recall is observed. From this it is clear that mining large amounts of unlabeled data to achieve good coverage is at least partially successful. Interestingly, it turned out that forward and backward preferences had similar precision, despite the fact that linguistically one might expect forward preferences to be more accurate. (Many queries are noun phrases, and English noun phrases are head-final.) It was the case that more forward preferences than backward preferences were found, and so forward preferences contribute somewhat more to recall. Note that, counterintuitively, precision for the selectional preference classifier drops slightly as the required recall is decreased to very low levels. This corresponds to using only the strongest selectional preferences in the classifier. Apparently the strongest predictors that words in the query belong to a category are not necessarily the strongest predictors that the query as a whole would be viewed as belonging to the category. We suspect that this anomaly can be fixed by more carefully choosing and combining measures of selectional preference strength.

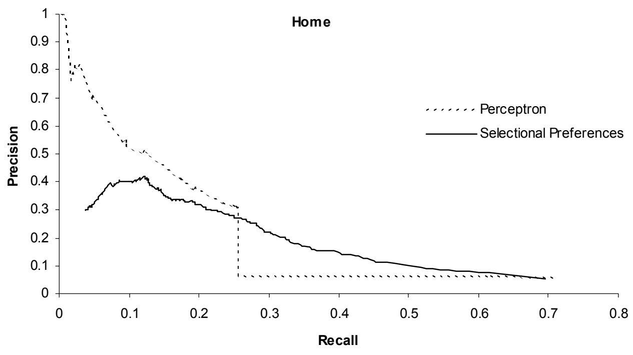

Figure 3: Precision & Recall for Perceptron & SP Classifiers on category "Home" at various thresholds

6. Conclusions & Future Work

We develop an approach for mining the vast amount of unlabeled data in web query logs and using it to improve automatic topical web query classification. We show that using our approach in combination with manual matching and supervised learning allows us to classify a substantially larger proportion of queries than any single technique. Moreover, our hope is that by leveraging unlabeled data we can minimize the need for periodically labeling new training data to keep up with changing trends in the query stream over time. There are several potential areas for future work. Of particular importance is the need to compare our selectional preferences approach with other methods of utilizing unlabeled data, such as EM and co-training. Others include expanding the initial seed-manual classification using queries classified by the framework, improving the linguistic properties of our SP algorithm, incorporating ideas from traditional rule-learning, examining the effectiveness of each approach on popular versus rare queries, and examining the performance of each approach on specific topical categories.

References

[1] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, A. Kolcz, O. Frieder, and D. Grossman, "Automatic Web Query Classification Using Labeled and Unlabeled Training Data," presented at SIGIR-2005, Salvador, Brazil, 2005.

[2] S. Beitzel, E. Jensen, A. Chowdhury, D. Grossman, and O. Frieder, "Hourly Analysis of a Very Large Topically Categorized Web Query Log," presented at ACM-SIGIR, 2004.

[3] D. Sullivan, "Searches Per Day," vol. 2003: Search Engine Watch, 2003.

[4] B. Jansen, A. Spink, and T. Saracevic, "Real life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web.," *Information Processing & Management*, vol. 36, pp. 207-227, 2000.

[5] L. Gravano, V. Hatzivassiloglou, and R. Lichtenstein, "Categorizing Web Queries According to Geographical Locality," presented at ACM CIKM, 2003.

[6] I.-H. Kang and G. Kim, "Query Type Classification for Web Document Retrieval," presented at ACM SIGIR, 2003.

[7] G. Salton, C. S. Yang, and A. Wong, "A Vector-Space Model for Automatic Indexing," *Communications of the ACM*, vol. 18, pp. 613-620, 1975.

[8] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," presented at ACM-SIGMOD, 2000.

[9] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Query Clustering Using User Logs," *ACM Transactions on Information Systems*, vol. 20, 2002.

[10] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Query Clustering Using Content Words and User Feedback," presented at ACM SIGIR, 2001.

[11] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering User Queries of a Search Engine," presented at WWW 2001, 2001.

[12] W. Krauth and M. Mezard, "Learning Algorityms with Optimal Stability in Neural Networks," *Journal of Physics A*, vol. 20, pp. pp. 745--752, 1987.

[13] L. L. Kupper and K. B. Hafner, "How Appropriate are Popular Sample Size Formulas?" *The American Statistician*, vol. 43, pp. pp. 101-105, 1989.

[14] C. D. Manning and H. Schutze, *Foundations of Statitical Natural Language Processing*: MIT Press, 1999.

[15] D. McCarthy and J. Carroll, "Disambiguating Nouns, Verbs, and Adjectives using Automatically Acquired Selectional Preferences," *Computational Linguistics*, vol. 29, pp. pp. 639--654, 2003.

[16] P. Resnik, "Selection and Information: A Class-Based Approach to Lexical Relationships," University of Pennsylvania, 1993.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*: Wiley, 1991.

[18] M. Light and W. Greiff, "Statistical Models for the Induction and Use of Selectional Preferences," *Cognitive Science*, vol. 87, pp. 1-13, 2002.

[19] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[20] D. D. Lewis, "Evaluating and Optimizing Autonomous Text Classification Systems," presented at SIGIR 1995, Seatlle, WA, 1995.

[21] C. J. v. Rijsbergen, *Information Retrieval*, 2nd ed. London: Butterworths, 1979.

[22] J. M. Tague, "The Pragmatics of Information Retrieval Experimentation," in *Information Retrieval Experiment*, K. S. Jones, Ed. London: Butterworths, 1981, pp. pp. 59-102.