Improving Web Search Relevance with Semantic Features

Yumao Lu Fuchun Peng Gilad Mishne Xing Wei Benoit Dumoulin
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA, 94089
yumaol,fuchun,gilad,xwei,benoitd@yahoo-inc.com

Abstract

Most existing information retrieval (IR) systems do not take much advantage of natural language processing (NLP) techniques due to the complexity and limited observed effectiveness of applying NLP to IR. In this paper, we demonstrate that substantial gains can be obtained over a strong baseline using NLP techniques, if properly handled. We propose a framework for deriving semantic text matching features from named entities identified in Web queries; we then utilize these features in a supervised machine-learned ranking approach, applying a set of emerging machine learning techniques. Our approach is especially useful for queries that contain multiple types of concepts. Comparing to a major commercial Web search engine, we observe a substantial 4% DCG5 gain over the affected queries.

1 Introduction

Most existing IR models score documents primarily based on various term statistics. In traditional models—from classic probabilistic models (Croft and Harper, 1979; Fuhr, 1992), through vector space models (Salton et al., 1975; Narita and Ogawa, 2000), to well studied statistical language models (Ponte and Croft, 2000; Lafferty and Zhai, 2001)—these term statistics have been captured directly in the ranking formula. More recently, *learning to rank* approaches to IR (Friedman, 2002) have become prominent; in these frameworks, that aim at learning a ranking function from data, term statistics are often modeled as *term matching features* in a machine learning process.

Traditional text matching features are mainly based on frequencies of n-grams of the user's query in a variety of document sections, such as the document title, body text, anchor text, and so on. Global information such as frequency of term or term group in the corpus may also be used, as well as its combination with local statistics – producing relative scores such as $tf \cdot idf$ or BM25 scores (Robertson et al., 1995). Matching may be restricted to certain window sizes to enforce proximity, or may be more lenient, allowing unordered sequences and nonconsecutive sequences for a higher recall.

Even before machine learning was applied to IR, NLP techniques such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and parsing have been applied to both query modeling and document indexing (Smeaton and van Rijsbergen, 1988; Narita and Ogawa, 2000; Sparck-Jones, 1999). For example, statistical concept language models generalize classic n-gram models to concept n-gram model by enforcing query term proximity within each concept (Srikanth and Srihari, 2003). However, researchers have often reported limited gains or even decreased performance when applying NLP to IR (Voorhees, 1999).

Typically, concepts detected through NLP techniques either in the query or in documents are used as proximity constraints for text matching (Sparck-Jones, 1999), ignoring the actual concept type. The machine learned approach to document ranking provides us with an opportunity to revisit the manner in which NLP information is used for ranking. Using knowledge gained from NLP application as features rather than heuristically allows us much greater flexibility in the amount and variability of information used – e.g., incorporating knowledge about the actual entity types. This has several benefits: first, entity types appearing in queries are an indicator of the user's intent. A query consisting of a business *category* and a location (e.g., *hotels Palo Alto*) appears to be

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 648–657,
Singapore, 6-7 August 2009.

informational, and perhaps is best answered with a page containing a list of hotels in Palo Alto. Queries containing a business *name* and a location (e.g., *Fuki Sushi Palo Alto*) are more navigational in nature – for many users, the intent is finding the home page of a specific business. Similarly, entity types appearing in documents are an indicator of the document type. For example, if "Palo Alto" appears ten times in document's body text, it is more likely to be a local listing page than a home page. For the query *hotels Palo Alto*, a local listing page may be a good page, while for the query *Fuki Sushi Palo Alto* a listing page is not a good page.

In addition, knowledge of the particular entities in queries allows us to incorporate external knowledge about these entities, such as entity-specific stopwords ("inc." as in *Yahoo Inc.* or "services" as in *kaiser medical service*), and so on.

Finally, even when using named entities only for deriving proximity-related features, we can benefit from applying different levels of proximity for different entities. For example, for entities like cities (e.g., "River Side"), the proximity requirement is fairly strict: we should not allow extra words between the original terms, and preserve their order. For other entities the proximity constraint can be relaxed—for example, for person names, due to the middle name convention: *Hillary Clinton* vs. *Hillary R. Clinton*.

In this paper, we propose a systematic approach to modeling semantic features, incorporating concept types extracted from query analysis. Vertical attributes, such as city-state relationships, metropolitan definition, or idf scores from a domain specific corpus, are extracted for each concept type from vertical database. The vertical attributes, together with the concept attributes, are used to compose a set of semantic features for machine learning based IR models. A few machine learning techniques are discussed to further improve relevance for subclass of difficult queries such as queries containing multiple types of concepts. Figure 1 shows an overview of our approach; after discussing related work in Section 2, we spend Sections 3 to 5 of the paper describing the components of our system. We then evaluate the effectiveness of our approach both using general queries and with a set of "difficult" queries; our results show that the techniques are robust, and particularly effective for this type of queries. We conclude in Section 7.

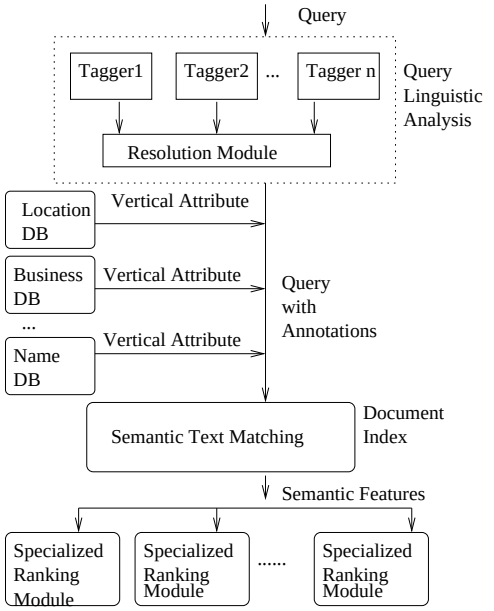

Figure 1: Ranking with Semantic Features

2 Related Work

There is substantial body of work involving usage of NLP techniques to improve information retrieval (Brants, 2003; Strzalkowski et al., 1996). Allan and Ragahavan (Allan and Raghavan, 2002) use Part-of-Speech tagging to reduce ambiguity of difficult queries by converting short queries to questions. In other POS-tags work, Arampatzis *et al.* (Arampatzis et al., 1990) observed an improvement when using nouns only for retrieval. Croft et al. (Croft et al., 1991) and Tong et al. (Buckley et al., 1993; Tong et al., 1996) explored phrases and structured queries and found phrases are effective in improving retrieval performance. Voorhees (Voohees, 1993) uses word sense disambiguation to improve retrieval performance. One IR domain that consistently benefits from usage of various NLP techniques is question answering, where queries are formed in natural language format; e.g., (Peng et al., 2005).

In general, however, researchers often observe limited gains or even degraded performance when applying NLP to IR (Voorhees, 1999). Having said this, most past studies use small datasets and a modest baseline; it is unclear whether a similar conclusion would be reached when using a state-of-art system such as a commercial web search engine as a baseline, and a full-web corpus – as we do in this paper. This leads to another difference between this work and existing work involving named entity recognition for retrieval. Most

previous research on usage of named entities in IR combines entity detection in documents and queries (Prager et al., 2000). Entity detection in document has a high indexing cost that is often overlooked, but cannot be ignored in the case of commercial search engines. For this reason, we restrict NLP processing to queries only – although we believe that document-side NLP processing will provide additional useful information.

3 Query Analysis

We begin by briefly describing our approach to named entity recognition in web queries, which serves as the basis for deriving the semantic text matching features.

Named entity recognition (NER) is the task of identifying and classifying entities, such as person names or locations, in text. The majority of state-of-the-art NER methods utilize a statistical approach, attempting to learn a mapping between a sequence of observations (words) and a sequence of tags (entity types). In these methods, the sequential nature of the data is often central to the model, as named entities tend to appear in particular context in text. For example, for most types of text, in the two sequences *met with X* and *buy the Y*, the likelihood of X being a person name is substantially higher than the corresponding likelihood of Y. Indeed, many named entity taggers perform well when applied to grammatical text with sufficient contexts, such as newswire text (Sang and Meulder, 2003).

Web queries, however, tend to be short, with most queries consisting of 1–3 words, and lack context – posing a particular challenge for identifying named entities in them. Existing work on NER in web queries focuses on tailoring a solution for a particular entity type and its usage in web search (Wang et al., 2005; Shen et al., 2008); in contrast, we aim at identifying a large range of possible entities in web queries, and using a generic solution for all of them.

In web queries, different entity types may benefit from different detection techniques. For example, an entity type with a large variability among instances as well as existence of external resources like product name calls for an approach that can make use of many features, such as a conditional random field; for entity types that are more structured like person names, a grammar-based approach can be more effective (Shen et al., 2008). To this end, we utilize multiple approaches for entity detection and combine them into a single, coherent "interpretation" of the query.

Given a query, we use several entity recognizers in parallel, one for each of the common entity types found in web queries. The modeling types may differ between the recognizers: some are Markovian models, while others are just dictionary lookups; the accuracy of each recognizer is also different. We then have a machine-learned disambiguation module that combines output from different taggers, ranking the tagging sequences. The details of scoring is out of the scope of this paper, and we omit it for simplicity.

4 Semantic Text Matching Features

Our proposed semantic features operate at the semantic type level rather than at the term level: instead of matching a term (or set of terms) in documents, we match their semantic type. Given the query *San Francisco colleges* and the annotation $[San\ Francisco]_{CityName}\ [colleges]_{BusinessCategory}$, the semantic text matching features would describe how relevant a document section is for a entity of type $CityName$, for $BusinessCategory$, and for their combination.

Concretely, we exploit a set of features that attempts to capture proximity, general relevance, and vertical relevance for each type of semantic tag and for each section of the document. We now review these feature by their broad types.

4.1 Semantic Proximity Features

Proximity features—features that capture the degree to which search terms appear close to each other in a document—are among the most important feature sets in ranking functions. Traditional proximity features are typically designed for all query terms (Metzler and Croft, 2005) and may suffer from wrong segmentations of the query. For example, for the query *New York city bus charter*, a traditional proximity feature may treat "city bus" similarly to "York city." But given detailed information about the entities in the query in their types, we can enforce proximity for "New York city" and "bus charter" more accurately. Different types of entities usually have different proximity characteristics in relevant documents. Strongly-bound entities such as city names typically have very high proximity in relevant documents, while entities such as business names may have much

lower proximity: a search for *Kaiser medical office*, for example, may be well-served with documents referring to *Kaiser Permanente medical office*, and as we mentioned before, person names matches may also benefit from lenient proximity enforcement. This is naturally addressed by treating each entity type differently.

We propose a set of semantic proximity features that associate each semantic tag type with generic proximity measures. We also consider tagging confidence together with term group proximity; we discuss these two approaches next.

4.1.1 Semantic Minimum Coverage (SMC)

Minimum Coverage (MC) is a popular span based proximity distance measure, which is defined as the length of the shortest document segment that cover the query term at least once in a document (Tao and Zhai, 2007). We extend this measure to Semantic Minimum Coverage (SMC) for each semantic type t in document section s and define it as

$$\text{SMC}_{t,s} = \frac{1}{|\{k|T_k = t\}|} \sum_{i \in \{k|T_k=t\}} w_i \text{MC}_{i,s},$$

where w_i is a weight for tagged term group i, $\text{MC}_{i,s}$ is the the minimum coverage of term group i in document section s, $\{k|T_k = t\}$ denotes the set of all concepts having type t, and $|\{k|T_k = t\}|$ is the size of the set. The definition of the weight w is flexible. We list a few candidate weighting schemes in this paper: uniform weights (w^u), weights based on idf scores (w^{idf}) and "strength"-based weight (w^s), which we define as follows:

$$w^u = 1;$$
$$w^{\text{idf}} = \frac{c}{f_q}$$

where c is a constant and f_q is the frequency of the term group in a large query log;

$$w^s = \min_l \text{MI}_l$$

where MI_l is the point-wise mutual information of the l-th consecutive pair within the semantic tag. We can also combine strength and idf scores such that the weight reflects both relative importance and constraints in proximity. In this paper, we use

$$w^{si} = w^s w^{\text{idf}}.$$

In Section 6, we use all four weighting schemes mentioned above in the semantic feature set.

4.1.2 Semantic Moving Average BM25 (SMABM25)

BM25, a commonly-used bag-of-words relevance estimation method (Robertson et al., 1995), is defined (when applied to document sections) as

$$\text{BM25} = \sum_j \text{idf}_j \frac{f_{j,s}(c_1+1)}{f_{i,s} + c_1(1 - c_2 + c_2 \frac{l_s}{\bar{l_s}})}$$

where $f_{j,s}$ is the frequency of term j in section s, l_s is the length of section s, $\bar{l_s}$ is the average length of document section s, c_1, c_2, c_3 are constants and the idf score of term j is defined as

$$\text{idf}_j = \log \frac{c_4 - d_j + c_5}{d_j + c_5},$$

where d_j is the number of sections in all collections that contains term j and c_4, c_5 are constants.

To characterize proximity, we could use a fixed length sliding window and calculate the average BM25. We further associate each sliding average BM25 with each type of semantic term groups. This results in a Semantic Moving Average BM25 (SMABM25) of type t, which we define as follows:

$$\frac{1}{|\{k|T_k = t\}|} \sum_{i \in \{k|T_k=t\}} (1/M) \sum_m BM25_m$$

where m is a fixed length sliding window m and M is the total number of sliding windows (that depends on the length of the section window size).

4.2 Semantic Vertical Relevance Features

Vertical databases contain a large amount of structured domain knowledge typically discarded by traditional web relevance features. Having access to the semantic types in queries, we can tap into that knowledge to improve accuracy. For example, term frequencies in different corpora can assist in determining relevance given an entity type. As we mentioned in Section 1, we observe that term frequency in a database of business names provides an indication of the business brand, the key part of the business name phrase. While both "yahoo" and "inc" are very common terms on the web, in a database of businesses only "inc" is common enough to be considered a stopword in the context of business names.

We propose a Vertical Moving Average BM25 (VMABM25) as a feature aiming at quantifying the vertical knowledge for web search. The basic idea here is to replace the idf score idf_j of

SMABM25 with an idf score calculated from a vertical database for type t, namely idf_j^t:

$$\frac{1}{|\{k|T_k = t\}|} \sum_{i \in \{k|T_k=t\}} (1/M) \sum_m \text{BM25}_{m,t}$$

where

$$\text{BM25}_{m,t} = \sum_j \text{idf}_j^t \frac{f_{j,s}(c1+1)}{f_{i,s} + c_1(1 - c_2 + c_2 \frac{l_s}{l_s})}$$

where the idf_j^t is associated with the semantic type t and calculated from the corpus associated with that type.

VMABM25 links vertical knowledge, proximity, and page relevance together; we show later that it is one of most salient features among all semantic features.

4.3 Generalized Semantic Features

Finally, we develop a generalized feature based on the previous features by removing tags. Semantic features are often sparse, as many queries contain one entity or no entities at all; generalized features increase their coverage by combining the basic semantic features. An entity without tag is essentially a segment.

A segment feature x_i for query i does not have entity type and can be expressed as

$$x_i = \frac{1}{K_i} \sum_{k=1}^{K_i} x_{T(k)}$$

where K_i is the number of segments in the query and $T(k)$ is the semantic type associated with kth concept.

Although these features are less informative than type-specific features, one advantage of using them is that they have substantially higher coverage. In our experiments, more than 40% of the queries have some identified entity. Another relatively subtle advantage is that segment features have no type related errors: the only possible error is a mistake in entity boundaries.

5 Ranking Function Optimization

The ultimate goal of the machine learning approach to web search is to learn a ranking function $h(x_i)$, where x_i is a feature vector of a query-document pair i, such that the error

$$L(h) \equiv \sum_{i=1}^{N} (y_i - h(x_i))^2 \qquad (1)$$

is minimized. Here, y_i is the actual relevance score for the query-document pair i (typically assigned by a human) and N is the number of training samples.

As mentioned in the previous Section, an inherent issue with semantic features is their sparseness. User queries are usually short, with an average length of less than 3 words. Text matching features that are associated with the semantic type of query term or term groups are clearly sparse comparing with traditional, non-entity text matching features – that can be derived for any query. When a feature is very sparse, it is unlikely that it would play a very meaningful role in a machine learned ranking function, since the error L would largely depend on other samples that do not contain the specific semantic features at all. To overcome the spareness issue and take advantage of semantic features, we suggested generalizing our features; but we also exploit a few ranking function modeling techniques.

First, we use a "divide-and-conquer" approach. Long queries usually contain multiple concepts and could be difficult to retrieve relevant documents. Semantic features, however, are rich in this set of queries. We may train special models to further optimize our ranking function for those queries. The loss function over ranking function h becomes

$$L_C(h) \equiv \sum_{i \in C} (y_i - h(x_i))^2 \qquad (2)$$

where C is the training set that falls into a predefined subclass. For example, queries containing both location and business name, queries contains both location and business category, etc, are good candidates to apply semantic features.

To this end, we first classify queries into several classes, each of which has multiple types of entities. The semantic features of those types would be dense for this subclass of queries. We then train models that may rank the specific class of queries well. This approach, however, may suffer from significantly less training samples due to training data partition resulted from the query classification. Increasing the modeling accuracy, then, comes at a cost of reduced data available for training. We apply two techniques to address this issue. The first approach is to over-weight subclass training samples such that the subclass of queries plays a more important role in modeling while still

keeping a large pool of the overall training samples. The second approach is model adaptation: a generalized incremental learning method. Here, instead of being over-weighted in a joint optimization, the subclass of training data is used to modify an existing model such that the new model is "adapted" to the subclass problem. We elaborate on our approaches as follows.

5.1 Weighted Training Samples

To take advantage of both large a pool of training samples and sparse related semantic features for a subclass of queries, we could modify the loss function as follows

$$L_C^w(h) \equiv w \sum_{i \in C} (y_i - h(x_i))^2 + \sum_{i \in \bar{C}} (y_i - h(x_i))^2, \quad (3)$$

where $\bar{C}$ is the complement of set C. Here, the weight w is a compromise between loss function (1) and (2). When $w = 1$, we have

$$L_C^1(h) \equiv L(h);$$

when $w -> \infty$

$$L_C^\infty(h) \equiv L_C(h).$$

A large weight may help optimize the training for a special subclass of queries, and a small weight may help to preserve good generality of the ranker. We could use cross-validation to select the weight w to optimize a the ranking function for a subclass of queries. In practice, a small w is desired to avoid overfitting.

5.2 Model Adaptation

Model adaptation is an emerging machine learning technique that is used for information retrieval applications with limited amount of training data. In this paper, we apply Trada, proposed by Chen et al. (Chen et al., 2008), as our adaptation algorithm.

The Trada algorithm aims at adapting tree-based models. A popular tree based regression approach is Gradient Boosting Trees (GBT) , which is an additive model $h(x) = \sum_{k=1}^{K} \gamma_k h_k(x)$, where each regression tree h_k is sequentially optimized with a hill-climbing procedure. As with other decision trees, a binary regression tree $h_k(x)$ consists of a set of decision nodes; each node is associated with a feature variable and a splitting value that partition the data into two parts, with the corresponding predicted value defined in the leave node. The basic idea of Trada is to apply piecewise linear transformation to the base model based on the new training data. A set of linear transformations are applied to each decision node, either predict or split point or both, such that the new predict or the split point of a node in a decision tree satisfies

$$v = (1 - p_C)\hat{v} + p_C v_C$$

where $\hat{v}$ denotes predict or split point of that node in the base mode and v_C denotes predict or split point of that node using new data set C, and the weight p_C depends on the number of original training data and new training data that fall through the node. For each node, the split or predict can be estimated by

$$p_C = \frac{\beta n_C}{n + \beta n_C},$$

where n is the number of training sample of the base model that fall through the node, n_C is the number of new training sample that fall through the node, and β is a parameter that can be determined using cross validation. The parameter β is used to over-weight new training data, an approach that is very effective in practice. For new features that are not included in the base model, more trees are allowed to be added to incorporate them.

6 Experiments

We now measure the effectiveness of our proposal, and answer related questions, through extensive experimental evaluation. We begin by examining the effectiveness of features as well as the modeling approaches introduced in Section 5 on a particular class of queries—those with a local intent. We proceed by evaluating whether if the *type* associated with each entity really matters by comparing results with type dependent semantic features and segment features. Finally, we examine the robustness of our features by measuring the change in the accuracy of our resulting ranking function when the query analysis is wrong; we do this by introducing simulated noise into the query analysis results.

6.1 Dataset

Our training, validation and test sets are human-labeled query-document pairs. Each item in the

sets consists of a feature vector $\mathbf{x_i}$ representing the query and the document, and a judgment score y_i assigned by a human. There are around 600 features in each vector, including both the newly introduced semantic features and existing features; features are either query-dependent ones, document-dependent ones, or query-and-document-dependent features.

The training set is based on uniformly sampled Web queries from our query log, and top ranked documents returned by commercial search engines for these queries; this set consists of 1.24M query-document pairs.

We use two additional sets for validation and testing. One set is based on uniformly sampled Web queries, and contains 42790 validation samples and 70320 test samples. The second set is based on uniformly sampled *local queries*. By local queries, we mean queries that contain at least two types of semantic tags: a location tag (such as street, city or state name) and a business tag (a business name or business category). We refer to this class of queries "local queries," as users often type this kind of queries in local vertical search. The local query set consists of 11040 validation samples and 39169 test samples. In the training set we described above, there are 56299 training samples out of the 1.24M total number of training samples that satisfy the definition of local queries. We call this set local training subset.

6.2 Evaluation Metrics

To evaluate the effectiveness of our semantic features we use Discounted Cumulative Gain (DCG) (Jarvelin and Kekalainen, 2000), a widely-used metric for measuring Web search relevance. (Jarvelin and Kekalainen, 2000). Given a query and a ranked list of K documents ($K = 5$ in our experiments), the DCG for this query is defined as

$$\mathrm{DCG}(K) = \sum_{i=1}^{K} \frac{y_i}{\log_2(1+i)}. \qquad (4)$$

where $y_i \in [0, 10]$ is a relevance score for the document at position i, typically assigned by a human, where 10 is assigned to the most relevant documents and 0 to the least relevant ones.

To measure statistical significance, we use the Wilcoxon test (Wilcoxon, 1945); when the p-value is below 0.01 we consider a difference to be statistically significant and mark it with a **bold font** in the result table.

6.3 Experimental Results

We use Stochastic Gradient Boosting Trees (SGBT) (Friedman, 2002), a robust none linear regression algorithm, for training ranking functions and, as mentioned earlier, Trada (Chen et al., 2008) for model adaptation.

Training parameters are selected to optimize the relevance on a separated validation set. The best resulting is evaluated against the test set; all results presented here use the test set for evaluation.

6.3.1 Feature Effectiveness with Ranking Function Modeling

We apply the modeling approaches introduced in Section 5 to improve feature effectiveness on "difficult" queries—those more than one entity type; we evaluate these approaches with the semantic-feature-rich set, the local query test set. We split training sets into two parts: one set belongs to the local queries, the other is the rest. We first weight the local queries and use the combined dataset as training data to learn the ranking functions; we train functions with and without the semantic features. We evaluate these functions against the local query test set. The results are summarized in Table 1, where w denotes the weight assigned to the local training set, bolded numbers are statistically significant result compared to the baseline, uniformly weighted training data without semantic features (with superscript b). It is interesting to observe that without semantic features, over-weighted local training data does not have statistically significant impact on the test performance; with semantic features, a proper weight over training samples does improve test performance substantially.

Table 1: Evaluation of Ranking Models Trained Against Over-weighted Local Queries with Semantic Features on the Local Query Test Set

	w/o semantic features		w/ semantic features	
Weight	DCG(5)	Impr.	DCG(5)	Impr.
$w = 0$	8.09^b	-	8.25	$\mathbf{2.0\%}$
$w = 2$	8.09	0.02%	8.26	$\mathbf{2.1\%}$
$w = 4$	8.13	0.49%	8.34	$\mathbf{3.1\%}$
$w = 8$	8.13	0.49%	8.42	$\mathbf{4.1\%}$
$w = 16$	8.13	0.49%	8.30	$\mathbf{2.6\%}$
$w = 32$	8.04	-0.60%	8.27	$\mathbf{2.2\%}$

Next, we use the local query training set as "new data" in the tree adaptation approach. In tree adaptations, all parameters are set to optimize the performance over the local validation set. We com-

pare two major adaptation approaches proposed in (Chen et al., 2008): adapting predict only and adapting both predict and split. We use the model trained with the combined training and uniform weights as the baseline; results are summarized in Table 2.

Table 2: Trada Algorithms with Semantic Features on Local Query Test Set

	w/o semantic feat.		w/ semantic feat.	
Ada. Appr.	DCG(5)	Impr.	DCG(5)	Impr.
Combined data	8.09^b	-	8.25	$\mathbf{2.0}\%$
Ada. predict	8.02	-0.1%	8.14	0.6%
Ada. predict & split	8.00	-0.1%	8.17	$\mathbf{1.0}\%$

Comparing Tables 1 and 2, note that using the combined training data with local query training samples over-weighted achieves better results than tree adaption. The latter approach, however, has the advantage of far less training time, since the adaptation is over a much smaller local query training set. With the same hardware, it takes just a few minutes to train an adaptation model, while it takes days to train a model over the entire, combined training data. Considering that massive model validation tasks are required to select good training parameters, training many different models with over a million training samples becomes prohibitly costly. Applying tree adaptation techniques makes research and prototyping of these models feasible.

6.3.2 Type Dependent Semantic Features vs. Segment Features

Our next experiment compares type-dependent features and segment features, evaluating models trained with these features against the local query test set. No special modeling approach is applied here; results are summarized in Table 3. We observe that by using type-dependent semantic features only, we can achieve as much as by using all semantic features. Since segment features only convey proximity information while the base feature set already contain a systematic set of proximity measures, the improvement through segment features is not as significant as the the type dependent ones.

6.3.3 Robustness of Semantic Features

Our final set of experiments aims at evaluating the robustness of our semantic features by introducing

Table 3: Type-dependent Semantic Features vs. Segment Features

Feature set	DCG(5)
base + type dependent semantic features	8.23
base + segment features	8.19
base + all semantic features	8.25

simulated errors to the output of our query analysis. Concretely, we manipulate the precision and the recall of a specific type of entity tagger, t, on the training and test set. To decrease the *recall* of type t, we uniformly remove a set of $a\%$ tags of type t – preserving precision. To decrease *precision*, we uniformly select a set of query segments (viewing the entity detection as simple segmentation, as detailed earlier) and assign the semantic type t to those segments. Since the newly added term group are selected from query segmentation results, the introduced errors are rather semantic *type* error than boundary error or proximity error. The total number of newly assigned type t tags are $b\%$ of the original number of type t tags in the training set. By doing this, we decrease the precision of type t while keeping the recall of it at the same level.

Suppose the original tagger achieves precision p and recall r. By removing $a\%$ of tags, we have estimated precision $\hat{p}$ and recall $\hat{r}$ defined as follows:

$$\hat{r} = \frac{100r - ar}{100},$$

$$\hat{p} = p.$$

By adding $b\%$ more term group to this type, we have estimated precision and recall as

$$\hat{p} = \frac{100p}{100 + bp},$$

$$\hat{r} = r.$$

In the experiment reported here we use BUSINESS NAME as the target semantic type for this robustness experiment. An editorial test shows that our tagger achieves 74% precision and 66% recall based on a random set of human labeled queries for this entity type. We train ranking models with various values of a and b. When we reduce the estimated recall, we evaluate these models against the local test set since other data are not affected. The results are summarized in Table 4.

When we reduce the precision, we evaluate the resulting models against the general test set as

Table 4: Relevance with simulated error on local query test set

a	b	$\hat{p}$	$\hat{r}$	DCG(5)	Impr.
0	0	0.74	0.66	8.25	–
10	0	0.74	0.594	8.21	0.48%
20	0	0.74	0.528	8.19	**0.72%**
40	0	0.74	0.396	8.18	**0.85%**

Table 5: Search relevance with simulated error for semantic features on general test set

a	b	$\hat{p}$	$\hat{r}$	DCG(5)	Impr.
0	0	0.74	0.66	10.11	-
0	10	0.689	0.66	10.11	0.00%
0	20	0.645	0.66	10.12	0.10%
0	40	0.571	0.66	10.12	0.10%
0	60	0.513	0.66	10.12	0.10%
0	80	0.465	0.66	10.11	0.00%
0	100	0.425	0.66	10.10	−0.10%

simulated errors would virtually affect any samples with certain probability. Results appear in Table 5. The results are quite interesting: when the recall of business name entity decreases, we observe statistically significant relevance degradation: if less entities are discovered, search relevance is hit. The experiments with simulated precision error, however, are less conclusive. One may note the experiments are conducted over the general test set. Therefore, it is not clear if the precision of the NER system really has insignificant impact on the IR relevance or just the impact is diluted in a larger test set.

6.4 Case Analysis

In this section, we take a close look at a few cases where our new semantic features help most and where they fail. For the query *silverado ranch in irving texas*, with no semantic features, the ranking function ranks a local listing page for this business, `http://local.yahoo.com/info-28646193`, as the top document. With semantic features, the ranking function ranks the business home page: `http://www.silveradoranchparties.com/` as top URL. Examining the two documents, the local listing page actually contains much more relevant anchor text, which are the among the most salient features in traditional ranking models. The home page, however, contains almost no relevant anchor text: for a small business home page, this is not a rare situation. Looking at the semantic features of these two pages, the highest resolution of location, the city name "Irving," appears in the document body text 19 times in the local listing page body text, and only 2 times in the home page body text. The training process learns, then, that for a query for a local business name (rather than a business category), home pages—even with fewer location terms in them—are likely to be more relevant than a local listing page that usually contain high frequency location terms.

In some cases, however, our new features do hurt performance. For the query *pa treasurers office*, the ranking function with no semantic features ranks the document `http://www.patreasury.org` highest, while the one with semantic features ranks the page `http://www.pikepa.org/treasurer.htm` higher. The latter page is somewhat relevant: it is a treasurer's office in Pennsylvania. However, it belongs to a specific county, which makes it less relevant than the former page. This is a classic error that we observe: a mismatch of the intended location area. While users are looking for state level business, we provide results of county level. To resolve this type of error, query analysis and semantic text matching are no longer enough: here, the ranking function needs to know that Pike County is a county in Pennsylvania, Milford is a city in Pike County, and neither are referred to by the user. Document-side entity recognition, however, may provide this type of information, helping to address this type of errors.

7 Conclusion and Future Research

In this paper, we investigate how semantic features can improve search relevance in a large-scale information retrieval setting; to our knowledge, it is the first study of this approach on a web scale. We present a set of features that incorporate semantic and vertical knowledge into the retrieval process, propose techniques to handle the sparseness problem for these features, and describe how they fit in the learning process. We demonstrate that these carefully designed features significantly improve relevance, particularly for difficult queries – long queries with multiple entities.

The work reported here focuses on query-side processing, avoiding the indexing cost of document processing. We are currently investigating document-side analysis to complement the query-side work, and believe that this will further boost the retrieval accuracy; we hope to report on this in a follow-up study.

References

J. Allan and H. Raghavan. 2002. Using Part-of-Speech Patterns to Reduce Query Ambiguity. In *Proceedings of SIGIR*.

A. T. Arampatzis, Th. P. Weide, C. H. A. Koster, and P. Bommel. 1990. Text Filtering using Linguistically-motivated Indexing Terms. Technical Report CSI-R9901, Computing Science Institute, University of Nijmegen, Nijmegen,The Netherlands.

Thorsten Brants. 2003. Natural Language Processing in Information Retrieval. In *Proceedings of CLIN*.

C. Buckley, J. Allan, and G. Salton. 1993. Automatic Routing and Ad-hoc Retrieval Using SMART: TREC 2. In *Proceedings of TREC-2*.

Keke Chen, Rongqing Lu, C.K. Wong, Gordon Sun, Larry Heck, and Belle Tseng. 2008. Trada: tree based ranking function adaptation. In *Proceedings of CIKM*.

W.B. Croft and D.J. Harper. 1979. Using Probabilistic Models of Document Retrieval without Relevance Information. *Journal of Documentation*, 37:285–295.

W. Bruce Croft, Howard R. Turtle, and David D. Lewis. 1991. The Use of Phrases and Structured queries in Information Retrieval. In *Proceedings of SIGIR*.

J. H. Friedman. 2002. Stochastic Gradient Boosting. *Computational Statistics and Data Analysis*, 38(4):367–378.

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35:243–255.

K. Jarvelin and J. Kekalainen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of SIGIR*.

J. Lafferty and C. Zhai. 2001. Document Language Models, Query Models and Risk Minimization for Information Retrieval. In *Proceedings of SIGIR*.

Donald Metzler and W. Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of SIGIR*.

M. Narita and Y. Ogawa. 2000. The Use of Phrases from Query Texts in Information Retrieval. In *Proceedings of SIGIR*.

Fuchun Peng, Ralph Weischedel, Ana Licuanan, and Jinxi Xu. 2005. Combining Deep Linguistics Analysis and Surface Pattern Learning: A Hybrid Approach to Chinese Definitional Question Answering. In *In Proceedings of the HLT-EMNLP*.

J. Ponte and W.B. Croft. 2000. A Language Modeling Approach to Informaiton Retrieval. In *Proceedings of SIGIR*.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. 2000. Question-answering by Predictive Annotation. In *Proceedings of SIGIR*.

Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. In *Proceedings of TREC-3*.

G. Salton, C. S. Yang, and C. T. Yu. 1975. A Theory of Term Importance in Automatic Text Analysis. *Journal of the Ameican Society of Information Science*, 26:33–44.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL*.

Dou Shen, Toby Walkery, Zijian Zheng, Qiang Yang, and Ying Li. 2008. Personal Name Classification in Web Queries. In *Proceedings of WSDM*.

A.F. Smeaton and C.J. van Rijsbergen. 1988. Experiments on Incorporating Syntactic Processing of User Queries into a Document Retrieval Stragegy. In *Proceedings of SIGIR*.

K. Sparck-Jones, 1999. *What is the Role of NLP in Text Retrieval*, pages 1–25. Kluwer.

M. Srikanth and R.K. Srihari. 2003. Incorporating Query Term Dependencies in Language Models for Document Retrieval. In *Proceedings of SIGIR*.

Tomek Strzalkowski, Jose Perez-Carballo, Jussi Karlgren, Anette Hulth, Pasi Tapanainen, and Timo Lahtinen. 1996. Natural Language Information Retrieval: TREC-8 Report. In *Proceedings of TREC-8*.

Tao Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of SIGIR*.

X. Tong, C. Zhai, N. Millic-Frayling, and D Evans. 1996. Evaluation of Syntactic Phrase Indexing – CLARIT NLP Track Report. In *Proceedings of TREC-5*.

E. Voohees. 1993. Using WordNet to Disambiguate Word Senses for Text Retrieval. In *Proceedings of SIGIR*.

Ellen Voorhees. 1999. Natural Language Processing and Information Retrieval. *Lecture Notes in Computer Science*, 1714:32–48.

Lee Wang, Chuang Wang, Xing Xie, Josh Forman, Yansheng Lu, Wei-Ying Ma, and Ying Li. 2005. Detecting dominant locations from search queries. In *Proceedings of SIGIR*.

F. Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics*, 1:80–83.