Concordance-Based Entity-Oriented Search

Mikhail Bautin
Department Of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
mbautin@cs.sunysb.edu

Steven Skiena
Department Of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
skiena@cs.sunysb.edu

***Abstract*— We consider the problem of finding the relevant named entities in response to a search query over a given text corpus. Entity search can readily be used to augment conventional web search engines for a variety of applications.**

To assess the significance of entity search, we analyzed the AOL dataset of 36 million web search queries with respect to two different sets of entities: namely (a) 2.3 million distinct entities extracted from a news text corpus and (b) 2.9 million Wikipedia article titles. The results clearly indicate that search engines should be aware of entities, for under various criteria of matching between 18-39% of all web search queries can be recognized as specifically searching for entities, while 73-87% of all queries contain entities.

Our entity search engine creates a concordance document for each entity, consisting of all the sentences in the corpus containing that entity. We then index and search these documents using open-source search software. This gives a ranked list of entities as the result of search. Visit `http://www.textmap.com` for a demonstration of our entity search engine over a large news corpus.

We evaluate our system by comparing the results of each query to the list of entities that have highest statistical juxtaposition scores with the queried entity. Juxtaposition score is a measure of how strongly two entities are related in terms of a probabilistic upper bound. The results show excellent performance, particularly over well-characterized classes of entities such as people.

I. Introduction

We consider the challenge of building a search engine that retrieves appropriate entities (e.g. people, places, things) in response to user queries over a web-scale text corpus containing these entities. Current search engines return only documents or webpages in response to user queries instead of entities. We believe it would be quite valuable for users to get lists of the "most relevant" entities to their query in addition to the most relevant documents.

Concrete applications of entity search are detailed below, but we begin with some motivational examples of our search engine in action. Table I shows the top five results returned by our system for six queries. Three of these queries are entities themselves ("NEW YORK YANKEES", "GOOGLE" and "TENNIS") and the top result returned for them is the query itself. The other results returned for entity queries are connected with the queried entity in various ways: famous tournaments and players for tennis; its CEO name, headquarters location and rival search engine companies for Google; and the names of the manager, players and a rival team for the New York Yankees. The other three queries ("POLITICAL CORRUPTION", "POLARIZING FIGURE", and "BRITISH PRIME MINISTER") are not entities but concepts indirectly referring to entities. The results for the "POLITICAL CORRUPTION" query are three politicians investigated for corruption, one involved in a violation of ethic rules, and the controversial King of Nepal. The "POLARIZING FIGURE" returns the names of four politicians inspiring strong but differing opinions. The "BRITISH PRIME MINISTER" query returns three former and the current Prime Minister of the United Kingdom. We encourage the reader to experiment with our search engine at `http://www.textmap.com`.

	TENNIS	NEW YORK YANKEES	GOOGLE
1	Tennis	New York Yankees	Google
2	Roger Federer	Joe Torre	Yahoo
3	Andre Agassi	Alex Rodriguez	Eric Schmidt
4	U.S. Open	Derek Jeter	Mountain View
5	Andy Roddick	Boston Red Sox	Microsoft

	POLITICAL CORRUPTION	POLARIZING FIGURE	BRITISH PRIME MINISTER
1	Jack Abramoff	Hillary Rodham Clinton	Tony Blair
2	George Ryan	Katherine Harris	Winston Churchill
3	Tom DeLay	David Geffen	Margaret Thatcher
4	Pete Domenici	Donald H. Rumsfeld	British
5	King Gyanendra	Dick Cheney	Gordon Brown

TABLE I
RESULTS FOR CERTAIN QUERIES

Compelling applications of entity-targeted search over unstructured text include:

- *Navigational Search.* Augmenting document results of a conventional Web search engine with entity results are particularly relevant for *navigational* queries [1]. The related entity results may help satisfy the user's information need, as per question answering systems. Otherwise they can provide meaningful navigational alternatives to user's document-oriented query.
  Indeed, we present experimental results demonstrating that our methods can predict roughly 5-10% of user's subsequent entity queries. This is a large enough fraction of user queries to justify displaying navigational shortcuts to speed search.
- *Encyclopedia Search.* When the text corpus is (or is comparable) to a collection of encyclopedia entries, the performance of article retrieval can be improved by taking

into account all mentions of an entity across articles, rather than just in the entry corresponding to that entity. We are confident that our techniques could be used to improve the performance of the Wikipedia search engine. For example, a search on Wikipedia for "Microsoft chairman" returns as the top result a stub article for Helmut Panke, who is a member of the Board of Directors of Microsoft and a former chairman of BMW AG. It lists Bill Gates as the 26th most relevant article for this query. Our system, in contrast, correctly returns Bill Gates at the top position.

- *Product Search.* Aggregating all mentions of specific products in reviews, blogs and webpages results in a higher recall than existing product search engines, many of which currently just search product names, or limited collections of product descriptions. In the terminology of [1], this can improve handling of *transactional* web queries.

The contributions of our work are as follows:

- *Analysis of entity occurrences in Web query logs.* The best way to understand how search queries should be preprocessed and answered is to analyze past query data [2]. By analyzing the 36 million queries of the AOL dataset [3], we found that 20-40% of web search queries consist solely of single entities, while 70-87.5% queries contain entities as part of them. These findings demonstrate that a very high percentage of all web queries recognizably target entities or have entities associated with them.
- *A first-in-literature implementation of an entity search engine.* We approach the entity retrieval problem by utilizing all occurrences of each specific entity throughout a text corpus. For every entity, we automatically compose a *concordance*—a document capturing the context of all the occurrences of the entity in the corpus. Then we index and search these documents using an open source information retrieval package (Lucene) with a scoring scheme customized to reflect the specificity of automatically generated documents.
  Although our prototype search engine has been developed over a modest-sized 18 GB corpus of news, we see no fundamental difficulties in scaling this to web-scale search.
- *Empirical evaluation of our search engine.* Identifying a gold standard to evaluate the performance of our "first-in-literature" entity search engine is a non-trivial task. We do so by comparing the top entities returned when the query is itself a single entity to the entities having the top statistical *juxtaposition score* [4] with the queried entity. Juxtaposition score is a measure of how much more frequently two entities co-occur than they would by chance. Our search engine provides a much greater flexibility in entity retrieval by allowing free-text queries than is possible using juxtaposition scores.
- *Time-dependent entity/document search.* The most relevant entities associated with a query evolve during time. *Jennifer Aniston* is now less relevant to *Brad Pitt* than *Angelina Jolie*, even though her total number of co-locations may exceed those of her rival. To account for new articles added every day to the corpus, and to weigh recent hits higher, we support generation of multiple separate concordances for the same entity over disjoint time periods (e.g. months) and aggregate the hits for all such periods into a single result.
  Through analysis of the AOL query dataset, we determine the optimal discounting of entity references over time to identify the most relevant entities at time of search.

The rest of this document is organized as follows. Section II reviews previous work on entity-aware search. Section III provides motivation for the use of entities in search engines by analyzing occurrences of news and Wikipedia based entities in a web query log. Section IV describes the design of our news entity search system. Section V provides an empirical evaluation of the retrieval performance of our system. Section VI concludes the report and outlines the directions of our future work.

II. Related Work

Our entity-search engine is built on top of our Lydia news analysis system [5], [6], [4], [7], [8]. The Lydia system automatically builds an entity database from online U.S. newspapers downloaded on a daily basis. The techniques used for entity identification include part-of-speech tagging, templates and gazetteers, as well as clustering for coreference resolution. We use the resulting text with marked-up entities and the entity database to provide search targeting the extracted entities.

The current research relevant to entity-aware search can be subdivided into the following directions: augmenting traditional document retrieval systems with the knowledge of entities [9], [10]; the Semantic Web research aiming to create a Web of interrelated entities and thus greatly simplify entity search and improve document search [11], [12]; extraction of relations between entities from unstructured text and searching these relations [13], [14].

We detail two papers most relevant to our work. Chu-Carroll, et al. [9] describe how XML fragment query language can be applied to semantic search. They use input text with named entities and relations marked up. Queries are allowed to specify a general semantic category in place of a term, restrict term meaning to a certain category, or specify a relation between queried entities. The paper targets applications where loss in recall is less important than high precision, such as intelligence investigations.

Carpenter [10] describes an attempt to use named entity recognition to improve search results. They compare the baseline Lucene *tf.idf*-based approach and the one that uses LingPipe named entity recognition to match phrases in the query.

III. Entities in Web Queries

To gain insight into the performance of search engines, it is essential to analyze past query logs [2]. Seeking motivation

for entity-oriented search we asked the question: how often do web search queries recognizably target entities?

Search queries contain highly proprietary information, and therefore search engine companies do not often make it available to researchers. Fortunately a comprehensive web query dataset became available to us in August 2006 when AOL unintentionally released 36 million search queries by 500,000 users collected over three months. Although this release represented a serious violation of user privacy [3], the dataset is very useful for collecting cumulative statistics on web queries. When analyzing it, we did not use the user ID field or manually examine individual low-frequency queries, thus maintaining and respecting user privacy.

A. Approach to Analyzing Web Queries

We chose the method of identifying entities by matching search queries to existing lists of known entities. Another possible approach might try to recognize named entities in query text using a statistical named entity recognizer such as LingPipe [15], but this would be much less accurate due to the lack of capitalization and contextual information in web query data. For comparative purposes, we also used approximately three million entities identified from entry titles in Wikipedia. The results were similar enough to the news entities that they have been omitted due to space concerns.

We were interested in both *perfect matches*, where the entire query is an entity from our database, and *partial matches*, where an entity is contained in the query as a contiguous range of tokens (a "sub-query"). Many of the partial matches became perfect matches if relaxed criteria for matching entities with queries were used, because of typos and word separation alternatives. Therefore, we considered the following levels of strictness of matching queries with entities:

- *Exact comparison.* The exact case-insensitive appearance of entity in the query is required.
- *Alias resolving comparison.* For every entity in the database and every query we construct a list of aliases. By alias we mean the original string with different tokenization, punctuation, prefixes and/or suffixes:
  - URL normalization. The "http://", "www." prefixes and ".com", ".net" etc. suffixes are removed.
  - "&" is replaced with "and";
  - "Inc.", "Co.", "Corp." suffixes are removed;
  - plural nouns are reduced to singular, etc.

  If at least one alias of an entity matches with at least one alias of a query, the query is considered an entity query. The same criteria is used for a "sub-query" (a contiguous range of tokens in the query) to locate partial matches.
- *Alias resolving with phonetic hashing.* To further explore possible entity appearances in search queries and to deal with misspellings, we add a Double Metaphone [16] hash of every entity, query and sub-query to its respective alias list.

All Queries

Matches	No aliases	Aliases	Metaphone
perfect	17.91%	26.50%	38.82%
partial	55.14%	53.59%	48.41%
total	73.05%	80.09%	87.23%

Unique Queries

perfect	2.07%	5.33%	18.57%
partial	68.85%	69.72%	65.23%
total	70.92%	75.05%	83.80%

TABLE II
MATCH FREQUENCIES FOR ALL 36,389,577 QUERIES AND 10,154,743 UNIQUE QUERIES (AFTER DUPLICATE REMOVAL) COMPARED AGAINST LYDIA ENTITY LIST.

B. Frequencies of News Entities in Queries

Table II presents the percentage of perfect and partial matches of AOL queries to Lydia entities under three levels of matching strictness discussed above. As the "no aliases" column in the "all queries" part of Table II indicates, almost 18% of queries exactly match one of the entities in our database. Interestingly, these queries constitute only 2% of all unique queries. Entities extracted from the news are inherently popular and likely to be searched for.

Moving to the "aliases" column in the "all queries" part of Table II, we see a 48% gain in the number of perfect matches, indicating that there are many mistyped variations of how entity names are formatted in search queries. The addition of metaphone hashing of queries increases the number of perfect entity matches by 46.5%. Clearly, spelling correction is an important problem in both entity-oriented and document-oriented search engine design.

To summarize, from Table II we see that 73%-87% queries contain part that is recognizable as an entity, and 18%-39% queries are completely recognizable as entity names.

Perfect Matches

No aliases		Aliases		Metaphone	
unknown	3741787	unknown	3538105	unknown	4347665
person	842244	website	1851898	title	4192761
website	641492	title	1672178	website	1949374
organization	268081	person	944648	person	1798689
name	237937	company	301037	company	274560
company	114401	name	193818	place	188644
disease	68631	organization	169132	name	169435
place	66809	place	102282	organization	151581
last name	59450	TV series	98717	TV series	116031
university	46352	disease	71247	movie	87783

TABLE III
MATCHES OF NEWS ENTITIES WITH QUERIES BY CATEGORY

1) Frequency of Entities in Queries by Category: The Lydia system's entity database has category information (place, person, city, country etc.) available for each entity. We use the taxonomy of categories described in [17]. This category information has been obtained using a naïve Bayes classifier trained on a 2-to-3 word context of entity occurrences in news,

so it is not always accurate. However, counting frequencies of search query categories provides some useful insight.

Table III shows category distribution corresponding to cells of Table II. Only top ten categories are shown for each experiment. Unknown represent the 30-60% of entities the naïve Bayes classifier was unable to assign categories to. Moving from the "no aliases" to "aliases" column in Table III much more websites and titles get recognized, proving the usefulness of our URL matching method. When we go from the "aliases" to "metaphone" column, we get much more title matches, indicating that a substantial number of misspellings in titles get recognized by using metaphone.

IV. Concordance-Based Entity Search

For the design of our entity search engine, we perform retrieval of entities based on all occurrences of each particular entity throughout a text corpus. For every entity we generate a "concordance"—a text document containing all unique sentences from the text corpus in which that entity occurs. Then we search these documents with an open-source search engine (Lucene). This approach allows us to leverage the existing development in document-oriented information retrieval.

A. Indexing

During the indexing phase, an entity index is constructed, which is later used by the search server to handle online queries. The indexing procedure starts with processing news articles with the Lydia pipeline. The next step separates input documents into sentences so that concordances can be built. Each operation is done in a scalable distributed way.

1) Processing News Articles with the Lydia Pipeline: We process news articles through the Lydia pipeline performing named entity recognition and coreference set identification, and obtain output in XML format with entities marked up with `<pn>` tags and a category assigned to each entity occurrence. For bulk processing of large amount of news, we run multiple instances of the Lydia pipeline using the Condor job scheduling system [18].

2) Dealing with Near-Duplicate Articles: The web contains a substantial fraction of duplicate and near-duplicate documents [19]. We deal with the duplicate and near-duplicate article problem by eliminating duplicate sentences.

To obtain the set of unique sentences, as well as for other tasks, we use the Hadoop open-source implementation [20] of Google's MapReduce distributed computation model [21]. The map function takes an XML news article as input and produces (MD5 hash, sentence) tuples as output for every sentence in the input article that contains at least one entity marked up with a `<pn>` tag. The reduce function takes (MD5 hash, list of sentences) as input and outputs (MD5 hash, first sentence). The output of this MapReduce job is a collection of files containing all unique sentences of the input corpus.

3) Collecting Context of Every Entity: To produce a searchable document for each entity, we collect all the sentences containing that entity and concatenate them together. This is again done by means of a MapReduce job that takes a collection of unique sentences as input. The map function produces a set of (entity$_i$, sentence) tuples for every sentence, where entity$_i$ goes over all distinct entities occurring in the sentence. The reduce function takes (entity, list of sentences) as input and adds a document with two fields (entity, concatenation of sentences) to a Lucene index as output.

To estimate the increase in the relative index size compared to conventional document indexing, we calculated $\sum_s n_e(s)len(s) / \sum_s len(s)$, where s is a sentence and n_e is the number of entities in it. This statistic accounts for the fact that sentence s gets included in $n_e(s)$ concordance documents, and equals to 2.6 on our corpus.

B. Searching

Suppose document d_i is a concatenation of all unique sentences from the input corpus that contain the entity e_i. As the result of steps described in Section IV-A, we get a Lucene [22] index of documents with fields (e_i, d_i).

Lucene's scoring scheme must be modified to meaningfully search these automatically generated documents. The scoring formula used in Lucene, assuming that we are searching a single field, giving equal weights to all query terms, and omitting factors irrelevant to document ranking, is the following:

$$score(q,d) = coord(q,d)lengthNorm(d)\sum_{t\in q} tf(t,d)idf(t)^2$$

In the default implementation of Lucene scoring [23] $lengthNorm(d) = \frac{1}{\sqrt{numTerms(d)}}$, where $numTerms(d)$ is the number of terms in the document d. We found out that with this type of document length normalization, the top results almost always turn out to be very short documents, where the few matching terms gain an enormously high weight. To compensate for this, we set $lengthNorm(d) \equiv 1$ regardless of the document d.

Entity	Score	Month
Muhammad Yunus	1.000	200610
Muhammad Yunus	0.649	200612
Grameen Bank	0.548	200610
Bangladesh	0.509	200610
Muhammad Yunus	0.428	200611
Nobel Peace Prize	0.363	200612
Grameen Bank	0.336	200612
Nobel Peace Prize	0.336	200610
Nobel	0.324	200610
Dhaka	0.313	200610
Bangladeshi	0.313	200610

Entity	Score
Muhammad Yunus	1.000
Grameen Bank	0.548
Bangladesh	0.509
Nobel Peace Prize	0.363
Nobel	0.324
Dhaka	0.313
Bangladeshi	0.313

TABLE IV
On the left—Lucene results for a "Muhammad Yunus" query indexed in a time-dependent manner. On the right—the same results aggregated so that $score(entity) = \max_i score(entity, month_i)$.

C. Time-Dependent Indexing and Search

News search appears different from other document search because the news is a continuous flow of text, and the reader's

interest is often focused on the recently added documents. This has implications on the design of news article and entity retrieval systems [24]. In particular, the impact of recent text on search results should be higher than that of older text. Similar phenomena hold (to a lesser extent) in general web document search.

To address the time dependency problem and the document size restriction problem, we create multiple Lucene documents for each entity, with each document containing the context of all occurrences of the entity in a particular time period (in our case, month). Every concordance document in this case has three fields: (entity, concordance, month). We can view the Lucene results for our time-dependent index as a list of (entity, month) pairs with associated scores. An example of such results is shown in Table IV. There are multiple ways of assigning scores to an entity e based on multiple scores assigned to $(e, month_i)$ pairs:

- Exponential decay: This expresses exponentially decreasing user interest in past news.

$$score(e) = \sum_{i=0}^{k} \exp(-\alpha(k-i)) score(e, month_i) \quad (1)$$

  where k is the index of the current month, assuming that 0 corresponds to the earliest month for which news are included in the index.
- Maximum value:

$$score(e) = \max_{i=0,\ldots,k} score(e, month_i) \quad (2)$$

  The advantage of this method is speed of computation: if n top-scoring entities are requested, the scan of an (entity, month) hit list returned by Lucene can be stopped once n unique entities have been seen. The disadvantage of this method is that entities that were once very popular score higher than entities that have had steady popularity without high peaks.

D. Modeling User Interest in an Entity

To gain more insight into weighting timed hits, we hypothesize that the number of web queries containing an entity expresses user interest in that entity as a function of time. We use the daily scale instead of monthly for these experiments because only three months of web query data are available to us. We try to predict daily entity frequency in queries using its frequency in the news using the following models:

- Exponential decay with window w (including $w = \infty$):

$$h_{\beta,w}(e,i) = \sum_{i=\max(0,d-w+1)}^{d} \exp(-\beta(d-i)) n(e,i) \quad (3)$$

  where e is an entity, d is the index of the current day and $n(e,i)$ is the frequency of the entity e in the news on day i.
- Historical mean:

$$h_{average}(e,i) = \frac{1}{d+1} \sum_{i=0}^{d} n(e,i) \quad (4)$$

- A convex combination of both:

$$h_{\lambda,\beta,w}(e,i) = (1-\lambda) h_{\beta,w}(e,i) + \lambda h_{average} \quad (5)$$

Then, we optimize model parameters to maximize the Pearson correlation between $h(e,i)$ and the actual frequency $q(e,i)$ of entity e in web queries on day i. A correlation of 0.275 is reached by (4). A higher correlation of 0.349 is reached by (5) when $\beta = 0.01$ and $\lambda = 0$, showing that (3) is a better model than (4). We use $\alpha = \beta_{optimal} \times 30.4$ (average days per month) in our scoring function (1).

The dependency of the correlation given by $h_{0,w}(e,i)$ on the window size w is shown in Figure 1. The optimal $w = 28$ suggests the usual span of user interest an entity after it is mentioned in the news. However, this model would not be suitable for search result scoring, because it discards old references to an entity which could undoubtedly be of interest to the user. Therefore, we use $w = \infty$ in our system.

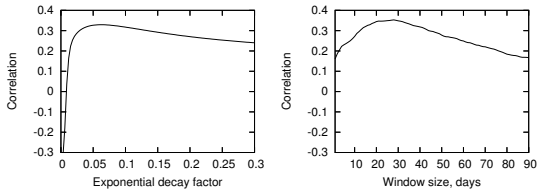

Fig. 1. The correlation between predicted and actual entity frequency in queries (l) depending on the exponential decay factor β for model (r) when predicted by summing entity frequency in the news on last w days.

V. Evaluation

Evaluating an entity search engine is a non-trivial problem. TREC (Text REtrieval Conference) provides a de facto standard to document retrieval and question answering systems evaluation. However, the problem of returning entities relevant to the user's query that we address here is different from both document retrieval and question answering problems.

Our Lydia text analysis system already contains a way to measure how strongly two entities are related to each other (*juxtaposition score*), expressed as an upper bound of the probability of them occurring in the same sentence under the assumption of independence [4]. Therefore, it would be natural for our search engine, when given a query that is a known entity itself, to return results close to the list of entities having top juxtaposition scores with the queried entity. The significant improvement of our search engine over the juxtaposition technique, however, is its ability to answer free text queries.

A. Comparison with Juxtaposition Lists

A method to identify entities that occur near a particular entity in an overrepresented way (i.e. more frequently than it would happen randomly) is described in [4]. Suppose, n_a and n_b are numbers of sentences containing entities a and b respectively, F is the number of sentences containing them both, and N is the total number of sentences in the corpus. Then, according to [4], the probability of the observed number

of occurrences under the assumption that these two entities are independent is not more than

$$P_{bound}(n_a, n_b, F, N) = \left(\frac{e^{\frac{FN}{n_a n_b}} - 1}{\left(\frac{FN}{n_a n_b}\right)^{\frac{FN}{n_a n_b}}} \right)^{\frac{n_a n_b}{N}} \quad (6)$$

We call $-\log$ of (6) the *juxtaposition score* of the entities a and b, meaning that the higher the juxtaposition score, the more dependency exists between the two entities. If for a given entity a we retrieve a list of $k-1$ entities $b_1, \ldots, b_{k-1}$ from our corpus that have highest juxtaposition scores with entity a, we will get a list of $k-1$ "most associated" with a entities in the sense of juxtaposition scores. It is natural to add the entity a itself onto the top of this list as being ultimately the most associated with itself. The resulting $(a, b_1, \ldots, b_{k-1})$ list is what we compare against the results of our search engine given the query a.

For comparison of top k lists we use the K_{min} distance measure described in [25]. According to [25], the K_{min} distance measure can be calculated as follows for two top k lists τ_1 and τ_2:

$$K_{min}(\tau_1, \tau_2) = K^{(0)}(\tau_1, \tau_2) = \sum_{\{i,j\} \subseteq D_{\tau_1} \cup D_{\tau_2}} \bar{K}^{(0)}_{i,j}(\tau_1, \tau_2)$$

where D_{τ_1} and D_{τ_2} are the sets of elements of τ_1 and τ_2 respectively, and $\bar{K}^{(0)}_{i,j}(\tau_1, \tau_2)$ is defined as as 0 if the elements i and j appear in the same order both in τ_1 and τ_2 and 1 if they appear in different order. An element that does not appear in one of the lists is considered appearing at the bottom of that list. If the elements i and j appear in one top k list but do not appear in the other top k list, $\bar{K}^{(0)}_{i,j}(\tau_1, \tau_2) = 0$.

To make these scores calculated for different list sizes comparable, we divide them by $\binom{k}{2}$ where k is the size of top lists. This corresponds to the value of the K_{min} distance measure between two top lists that contain the same k elements in the reverse order. When two lists are completely disjoint, this distance measure is equal to k^2.

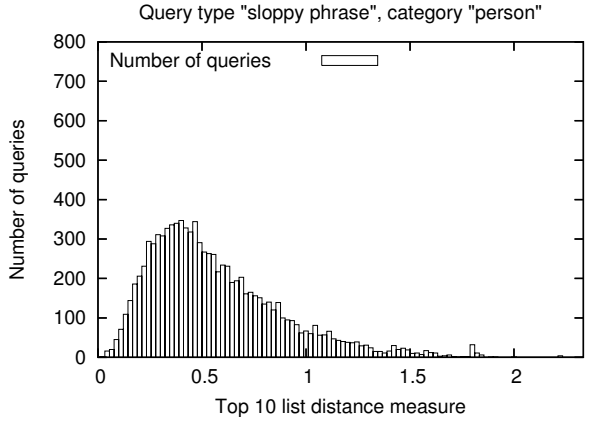

Fig. 2. Distribution of top list distances for phrase queries for the person category

	Phrase	Bag of words	Combination
Distance mean	0.831	1.154	1.148
Distance std dev	0.544	0.564	0.564

TABLE V
DISTANCE MEASURE BETWEEN SEARCH RESULTS AND JUXTAPOSITIONS FOR DIFFERENT QUERY TYPES FOR 9919 ENTITIES USED AS QUERIES.

1) Results by Entity Category: To determine how entity category effects search results, we have experimented with the top 10,000 entities in each category. The search results turn out to be the closest to the juxtaposition-based results for the "person" category, probably because Lydia provides more precise identification and categorization of "person" entities.

2) Results by Query Type: There are many possible ways to interpret an unstructured search query entered by the user:

- *"Bag of words" query*. In this case, the terms in the query are allowed to occur anywhere in the target document independently of each other, and a bonus score proportional to the number of appearing terms is given to each document.
- *"Phrase" query with different slop values*. The terms are required to appear in the document next to each other, but the order might differ from that specified by the user. The maximum edit distance where units correspond to movements of words should not exceed the slop value.
- *Combination of both "bag of words" and phrase queries.* We give the phrase query a significantly higher weight, so that when too few results are found for the phrase query, the results of the bag of words query are mixed in.

Table V shows the result of comparison of top-10 lists returned for about 10,000 most popular entities with top juxtaposition lists for the same entities. Of the three query types we used, the phrase query matches juxtapositions the best. In phrase queries in the Table V, and in the experiments in Section V-A.1, the slop value equal to the number of terms in the query is used.

3) Statistical Significance: Observe that if two result lists were independent, the probability of the top k lists drawn from N documents being completely disjoint would be

$$\frac{\binom{N-k}{k}}{\binom{N}{k}} \approx \left(\frac{N-k}{N}\right)^k \approx \exp\left(-\frac{k^2}{N}\right) \approx 0.9999 \quad (7)$$

with $k = 10$ and $N = 10^6$. Disjoint lists correspond to the distance measure of ≈ 2.222, but in any of our experiments at most 9500 queries out of 10000 have this distance measure. Therefore, the p-value of our observations is not more than $\sum_{k=0}^{9500} \binom{10000}{k} 0.9999^k 0.0001^{10000-k} < 10^{-100}$. This extremely low p-value can be explained by the fact that in our case the two top-k lists are not really independent at all, but are based on the same input data.

B. Query Prediction using Juxtapositions

Modern search engines suggest refined search queries to users, by (1) identifying likely spelling errors, and (2) analyz-

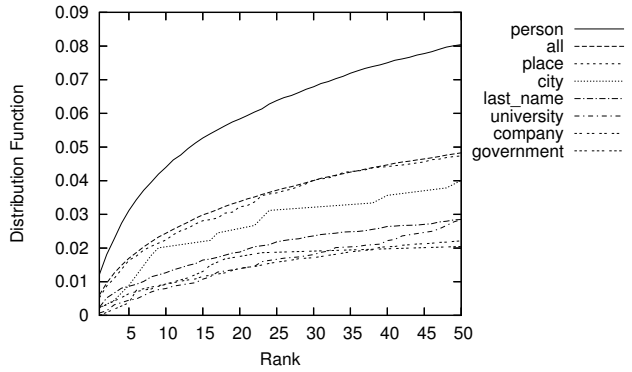

Fig. 3. Predicting the user's next query, i.e. the probability that an AOL search query ranks among the top k entities associated with the previous query (unique pair frequencies).

ing query frequencies in search logs by collaborative filtering. To augment these techniques, we propose using our entity search engine to suggest entities related to the previous search query as navigational links. The critical question is how often we can predict the user's next query from their current search.

We conducted preliminary experiments to get a handle on this. By analyzing the AOL query data set, we could identify pairs of successive queries by distinct users. (The IDs of all users remained anonymous through this process). We considered the 360,191 cases where users transitioned from one perfect entity query to a different perfect entity query. We were interested in the extent to which our search engine would report the latter entity in response to the first.

For efficiency of experimentation, we used the second entity's rank in the first entity's juxtaposition list as a proxy for the predictions of our search engine. Figure 3 presents our results for distinct pairs of perfect entity search queries as a function of rank position. It shows that our predictions are most accurate for people queries, predicting the next query over 8% of the time for rank=50. The decision procedure to refine the ranked list has not been optimized, and we anticipate substantially lower ranks will suffice for this level of predictability in practice.

These results are quite encouraging. They demonstrate that our methods can predict roughly 5-10% of subsequent user entity queries; enough to justify displaying our navigational shortcuts to aid their search.

VI. Conclusion and Future Work

It is becoming recognized that the new generation of search engines will need to be aware of entities in addition to searching unstructured text. We analyzed web query logs and found that up to 87% of web queries contain part that is recognizable as a news or Wikipedia entity. We then designed and implemented a prototype entity search engine that automatically composes a concordance capturing the context of all occurrences of each entity and leverages an off-shelf document retrieval technology (Lucene) to search these documents. The prototype is available at `http://www.textmap.com`.

Directions for future work include:

- Alternative evaluation techniques, possibly converting TREC list questions to queries to our search engine and finding the percentage of correct answers in the top list.
- Customizing our approach for product review search.
- Detection of target entity categories from the query and filtering results accordingly.

References

[1] A. Broder, "A taxonomy of web search," *SIGIR Forum*, vol. 36, no. 2, pp. 3–10, 2002.
[2] P. V. Dijck, "Better Search Engine Design: Beyond Algorithms," *O'Reilly ONLamp.com*, Aug. 2003.
[3] M. Arrington, "AOL Proudly Releases Massive Amounts of Private Data," http://www.techcrunch.com/2006/08/06, 2006.
[4] L. Lloyd, D. Kechagias, and S. Skiena, "Lydia: A system for large-scale news analysis." in *SPIRE*, 2005, pp. 161–166.
[5] N. Godbole, M. Srinivasaiah, and S. Skiena, "Large-Scale Sentiment Analysis for News and Blogs," in *Proceedings of International Conference on Weblogs and Social Media (to appear)*, Mar. 2007.
[6] J. H. Kil, L. Lloyd, and S. Skiena, "Question Answering with Lydia," in *The Fourteenth Text Retrieval Conference (TREC) Proceedings*, 2005.
[7] L. Lloyd, A. Mehler, and S. Skiena, "Identifying co-referential names across large corpora." in *CPM*, 2006, pp. 12–23.
[8] A. Mehler, Y. Bao, X. Li, Y. Wang, and S. Skiena, "Spatial Analysis of News Sources," in *IEEE Trans. Vis. Comput. Graph.*, vol. 12, 2006, journal, pp. 765–772.
[9] J. Chu-Carroll, J. Prager, K. Czuba, D. Ferrucci, and P. Duboue, "Semantic search via xml fragments: a high-precision approach to ir," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM Press, 2006, pp. 445–452.
[10] B. Carpenter, "Phrasal queries with lingpipe and lucene," in *Proceedings of the 13th Meeting of the Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, 2004.
[11] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.
[12] N. Shadbolt, B. T. Lee, and W. Hall, "The semantic web revisited," *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 21, no. 3, pp. 96–101, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1637364
[13] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web," in *IJCAI*, 2007.
[14] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain, "Names and similarities on the web: fact extraction in the fast lane," in *ACL '06*.
[15] "Alias-i Inc. LingPipe," http://www.alias-i.com/lingpipe.
[16] L. Phillips, "The Double Metaphone Search Algorithm," *C/C++ Users Journal*, June 2000.
[17] S. Sekine, K. Sudo, and C. Nobata, "Extended named entity hierarchy," in *Proceedings of the LREC-2002.*, 2002. [Online]. Available: citeseer.ist.psu.edu/sekine02extended.html
[18] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor – a distributed job scheduler," in *Beowulf Cluster Computing with Linux*, T. Sterling, Ed. MIT Press, October 2001.
[19] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin, "Query-free news search," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM Press, May 2003.
[20] Apache Software Foundation, "The Hadoop Project," http://lucene.apache.org/hadoop/.
[21] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," pp. 137–150. [Online]. Available: http://www.usenix.org/events/osdi04/tech/dean.html
[22] Apache Software Foundation, "Lucene," http://lucene.apache.org/.
[23] ——, "Apache Lucene—Scoring," http://lucene.apache.org/java/docs/scoring.html.
[24] G. M. D. Corso, A. Gullí, and F. Romani, "Ranking a stream of news," in *WWW '05: Proceedings of the 14th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2005, pp. 97–106.
[25] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2003. [Online]. Available: citeseer.ist.psu.edu/fagin03comparing.html