Data & Knowledge Engineering 90 (2014) 22–37

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Editorial

Wikimantic: Toward effective disambiguation and expansion of queries

Christopher Boston [a], Hui Fang [b], Sandra Carberry [a,*], Hao Wu [b], Xitong Liu [b]

[a] Department of Computer Science, University of Delaware, Newark, DE 19716, USA
[b] Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA

ARTICLE INFO

Article history:
Received 5 June 2013
Accepted 1 July 2013
Available online 15 August 2013

Keywords:
Disambiguation
Query expansion
Search queries

ABSTRACT

This paper presents an implemented and evaluated methodology for disambiguating terms in search queries and for augmenting queries with expansion terms. By exploiting Wikipedia articles and their reference relations, our method is able to disambiguate terms in particularly short queries with few context words and to effectively expand queries for retrieval of short documents such as tweets. Our strategy can determine when a sequence of words should be treated as a single entity rather than as a sequence of individual entities. This work is part of a larger project to retrieve information graphics in response to user queries.

1. Introduction

The amount of information available electronically has increased dramatically over the past decade. Research efforts in information retrieval and information extraction have developed methods for identifying documents relevant to a user query and for extracting information from such documents. These efforts have focused on textual documents and, to some extent, on pictorial images, but information graphics (non-pictorial graphics such as bar charts and line graphs) have been largely ignored. This is particularly problematic in the case of information graphics in popular media such as newspapers and magazines since the content of the graphic is generally not repeated in the article text [6]. This is in contrast with scientific articles where the article's text explicitly refers to and explains its constituent graphics. Thus information graphics are an important knowledge source that should not be ignored.

Information graphics in popular media generally have a high-level message that they are intended to convey, such as that *Visa ranks first among credit cards in circulation*. We have developed a methodology for identifying this high-level message via a Bayesian network that reasons about the content of the graphic and the graphic's communicative signals, such as one bar being colored differently from the other bars in a bar chart. [10,38,5]. We are now developing a system for retrieving information graphics in response to a user query by relating the query to a combination of the graphic's intended message and any text in the graphic such as the graphic's caption, axis labels, etc.

To do this, we must first disambiguate the words in the query and expand the query with related words. Words have multiple senses; for example, the word *bank* can refer to a financial institution or to the side of a river, depending on the context in which it is used. Disambiguation is the problem of determining the correct sense in which a word is used in a sentence. Disambiguation requires a knowledge source and a fundamental issue in disambiguation is the choice of the knowledge source; these have ranged from structured resources such as WordNet to unlabelled corpora [28]. Another issue is the segmenting of a word sequence into units that should be considered as a single concept. For example, should *comic book* be treated as a single unit or as two separate words? And lastly, one has the issue of which approach to take, ranging from supervised approaches that learn from labeled training sets to unsupervised methods which use unlabelled corpora [28].

* Corresponding author at: Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA. Tel.: +1 302 831 1954.
E-mail address: carberry@cis.udel.edu (S. Carberry).

0169-023X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.datak.2013.07.004

A vexing problem for information retrieval is the *term mismatch* problem — namely that queries often do not contain the same words that are used to index a document. For example, a query might use the word *company* whereas the document uses the term *corporation*. Query expansion is the task of expanding a query with related words that make for a more robust query that will be more successfully matched with relevant documents. As with disambiguation, query expansion requires a knowledge source and the selection of an approach, which can vary from linguistic approaches such as following the links in WordNet to statistical approaches that compute similarity based on occurrences within the same document [8]. In addition, query expansion does not seek a single interpretation, as is done in disambiguation, but instead must rank possible expansion terms and select a top-rated set of terms for inclusion in the expanded query.

Although disambiguation and query expansion are common issues in text processing and information retrieval, our problem is exacerbated by the fact that even full sentence queries for information graphics are short and the textual content of the graphics (including words provided by the graphic's hypothesized intended message) is small compared with that of typical text documents. For example, although there are many existing methods that extract semantic information via disambiguation, most require large amounts of context terms or focus exclusively on named entities [4,11,27,31].

This article presents our research on Wikimantic, a system initially designed for disambiguation of short queries and which has been extended to query expansion. By a short query, we mean one with fewer than 15 words, of which typically a third are non-content words such as prepositions, articles, and auxiliary verbs. In the case of query expansion, we focus on the problem of microblog retrieval, whose goal is to retrieve relevant tweets for a given topic [29]. This search domain is chosen for several reasons. First, tweets are much shorter than traditional documents and are thus similar in length to the limited text (caption, axis labels, and intended message) that will be available for a system designed to retrieve information graphics. Second, it is a difficult search domain. The short length of tweets makes retrieval more susceptible to failure since relevant tweets may not contain exactly the same terms as those appearing in the query. Third, microblog retrieval is an important search domain. With the increasing usage of Twitter, users have urgent needs to make sense and make use of the information hidden in the tweets. However, since this is a new search problem, it has not yet been well-studied.

Our approach both to disambiguation and query expansion utilizes the huge information resource provided by Wikipedia, the online encyclopedia hosted by the Wikimedia Foundation. Wikipedia has garnered a lot of interest as a tool for facilitating disambiguation by providing a semantic web of hyperlinks and “disambiguation pages” that associate ambiguous terms with unambiguous articles [15,25,27,31]. For an encyclopedia, English Wikipedia is monolithic. It contains over 3.5 million articles which are connected by hundreds of millions of user-generated links. Although errors do exist in articles and link structure, Wikipedia's strong editing community does a good job of keeping them to a minimum. A study [37] of the revision histories in Wikipedia showed that although malicious vandalism does occur, particularly with political articles, it is quickly repaired by constant communal editing. In addition, the rapid editing of existing articles and the construction of new ones is a strength of Wikipedia and results in an up-to-date knowledge source.

Our work has several novel contributions which are important for information systems. First, we disambiguate text strings that to our knowledge are the shortest yet. Second, our method can determine when a sequence of words should be disambiguated as a single entity rather than as a sequence of individual disambiguations. Furthermore, our method does not rely on capitalization since users are notoriously poor at correct capitalization of terms in their queries; this is in contrast to the text of formal documents where correct capitalization can be used to identify sequences of words that represent a named entity. With respect to query expansion, our method produces good results for retrieval of short documents such as tweets and outperforms all of the systems in the Microblog Track 2011 at TREC [29]. Thus our system Wikimantic offers promise not only for our research on retrieval of information graphics but more generally for information systems that must semantically process short text.

2. Related work

Bunescu and Pasca are generally credited with being the first to use Wikipedia as a resource for disambiguation [4]. They formulated the disambiguation task to be a two step procedure where a system must (1) identify the salient terms in the text and (2) link them accurately. Though Bunescu and Pasca's work was initially limited to named entity disambiguation, Mihalcea later developed a more general system that linked all “interesting” terms [25].

Mihalcea's keyword extractor and disambiguator relied heavily on anchor text extracted from Wikipedia's inter-article links. When evaluating the disambiguator, Mihalcea gave it 85 random Wikipedia articles with the linked terms identified but the link data removed, and scored it based on its ability to guess the original target of each link. Mihalcea achieved an impressive F-measure of 87.73 [25], albeit with one caveat. Regenerating link targets is significantly easier than creating them from scratch, since the correct target must necessarily exist in Wikipedia and be particularly important to the context. Wikimantic is tasked with the more difficult problem of disambiguating all salient terms in the query indiscriminately.

Many Wikipedia based disambiguation systems use variants of Mihalcea's method which attempt to match terms in the text with anchor text from Wikipedia links [27,18,24]. When a match is found, the term is annotated with a copy of the link. Sometimes, a term will match anchor text from multiple conflicting links, in which case the system must choose among them. Milne and Witten's contribution was to look for terms that matched only non-conflicting links, and use those easy disambiguations to provide a better context for the more difficult ones [27]. Given a large text string, it's always possible to find at least one trivial term to start the process. However, short strings do not reliably contain trivial terms.

Ferragina and Scaiella [15] addressed this problem by employing a voting system that resolved all ambiguous terms simultaneously. They found that good results were attainable with text fragments as short as 30 words each. Although their results

are very good, 30 words is still too large for the short queries we wish to process. In our evaluation, we limit our full sentence queries to a maximum of 15 terms in length. The average query length in our test set is just 8.9 words, including stop words.

Ratinov et al. define a local disambiguation method to be one that disambiguates each term independently, and a global disambiguation method to be one that searches for the best set of coherent disambiguations. Their recent work has shown that the best ranking performance can usually be obtained by combining local and global approaches [31]. Although their system was limited to named entities, our performance seems to be best when combining our own local and global approaches as well.

Abel et al. [1] focused on the problem of linking tweets with news articles and then enriching the tweets with the semantic concepts extracted from the content of the news articles. The semantic concepts were mainly entities, organizations or events. Their method tried to tag individual tweets with semantic concepts. On the contrary, we aim to disambiguate individual *words* or sequences of words from short texts.

More recently, Meij et al. studied the problem of linking search queries and tweets to Wikipedia concepts [21–23]. They formulated the problem as a supervised learning problem and solved it through a two-stage method. The first stage used all possible n-grams from a query or a tweet to retrieve relevant Wikipedia concepts as candidates. The retrieval process is based on Wikipedia titles, content and anchor text. In the second stage, a machine learning algorithm is then applied to rank all the candidate concepts based on multiple feature sets including n-gram features, concept features, twitter features or query log features. One advantage of our study is that we do not require any training data.

Query expansion refers to the issue of expanding a user query to include terms that are closely related to the terms that are given in the user query. For example, the user might include a word such as *company* in the query but a document might include the word *corporation*. The retrieval system should be able to handle such instances. Two methods for dealing with the problem are global query expansion and relevance feedback [20]. Global query expansion extracts synonyms of query terms from a source such as WordNet [13,7], Wikipedia [3,19,2], or via automatic thesaurus generation [16,35,40,26] and adds them to the user query. Relevance feedback [32,33] involves a cycle in which the top-rated relevant documents are used to produce a revised query that results in more effective document retrieval. Our work falls into the category of global query expansion.

Arguello et al. [3] expanded queries by first using the query to retrieve relevant Wikipedia articles. The K articles deemed most relevant were set aside. Anchor phrases were then selected from the next L most relevant articles, with the anchor phrases ranked based on how often they linked to one of the K most relevant articles. The highest ranked anchor phrases were then used to expand the query. Similarly Li et al. [19] expanded queries by first using the query to retrieve Wikipedia articles; Wikipedia categories were then assigned a weight equal to the number of retrieved articles in that category, articles were scored based on the weights of their categories, and 40 query expansion terms were extracted from the articles with the highest scores. Al-Shboul and Myaeng [2] attacked query expansion in the context of a patent search. Given a patent that is the basis for search, keywords were extracted to form the initial query. Using these keywords, Wikipedia articles were retrieved and scored based on primary Wikipedia categories, categories linked to the pages of primary categories, and page titles; expansion words were then selected from the top-ranked documents, categories and titles as well as from WordNet synsets. These methods rely on anchor phrases and Wikipedia categories and differ substantially from our approach, and none of these efforts target the retrieval of short documents via short queries.

Milne et al. [26] implemented query expansion via user interaction by using Wikipedia to automatically generate a corpus-dependent thesaurus. Each term in the document collection was associated with one or more Wikipedia articles; the term's correct sense was identified by measuring its semantic relatedness to the other terms in the sentence, paragraph, or entire document, where semantic relatedness of two terms was computed based on the probability of links common to the Wikipedia articles representing each term's possible senses. Terms in the document collection that were associated with the same Wikipedia article were entered into the thesaurus as synonyms. The user interface displayed terms from the thesaurus that were equated with the query terms, so that the user could select from among them to expand the query or to browse the thesaurus. Milne's work differs from ours in that the entirety of a long document can be used as the context for disambiguating the document's terms whereas we will be disambiguating terms in short queries and applying our methods to the retrieval of objects that contain only relatively short text.

3. Overview of paper

The remainder of this paper presents our research on Wikimantic which uses Wikipedia as the knowledge source for disambiguating terms in queries and for expanding words in queries. Given a sequence of terms, the disambiguation task consists of finding a mapping from each salient term in the sequence to the Wikipedia article that best represents the term in context. More formally, let $s = (t_1,t_2, \ldots,t_{|s|})$ be a sequence of $|s|$ terms. For every term t_j, if t_j is salient (not a function word), we wish to generate a mapping $t_j \rightarrow C_i$ where C_i is the Wikipedia article that best defines the concept t_j referenced. For example, given the sentence "Steve Jobs resigns from Apple", an acceptable mapping would link "steve" and "jobs" to the Wikipedia article about Steve Jobs, the former CEO of Apple. "resigns" would be mapped to the article Resignation, and "apple" would be mapped to the article for Apple Inc. Mapping "apple" to the article for the actual apple fruit would be unacceptable in that context.

Our general strategy is to first summarize the meaning of the sequence of terms s, which we will refer to as a *topic model* for s, and then use that summary to choose the most probable mapping of terms to concepts. To summarize s, we construct a "Concept"[1] object

[1] In this paper, we follow the convention that object types from our model are written in upper camel case. When we write "Concept", we refer specifically to the class of object from our model. When we write "concept", we are simply using the term as one would in every day speech.

that represents the general topic of *s*. The way we define and use Concept objects is based on our generative model as described in Section 4. Specifically, we begin by naively building a set of all Wikipedia articles that could possibly be referenced by terms in *s*. We weight each article with the product of its prior probability of being relevant and the degree to which terms in *s* match terms in the article. The weighted set is packaged up in a data structure we call a MixtureConcept. Section 4 presents the Wikimantic approach to constructing a topic model for a sequence *s* of terms, where the topic model consists of a set of weighted Concepts capturing the possible meanings of the terms in *s*.

Once we have the weighted set (the MixtureConcept), the individual terms in the sequence *s* can be disambiguated. This involves searching for the best mapping from terms in *s* to Wikipedia articles. We score each possible mapping according to the weights of the articles in the set. The details of this process are presented in Section 5, which also includes an evaluation of the Wikimantic approach to disambiguation.

The weighted set of Wikipedia articles can also be used to expand the sequence *s* with related terms. Section 6 presents our methodology for extending Wikimantic to query expansion, applies it to microblog retrieval, and presents results that demonstrate the success of the method. Section 7 discusses optimization issues and Section 8 concludes with a general summary and suggestions for future work.

4. Methodology for constructing topic models

In the field of information retrieval, the language modeling approach treats documents as generating queries and thus ranks documents for retrieval based on the probability of the query being generated by the document [20]. Such a model is often referred to as a generative model. Similarly, we can view ideas or concepts as generating words. An author encodes ideas into words and puts the words on paper. A reader may later take these words and decode them back into ideas. Our generative model is based on the premise that every idea has certain associated words that are used to talk about the idea. A person writing about the Apple Corporation may use terms like “computer”, “iPhone”, “Steve”, or “Jobs”. A reader can use a priori knowledge about these term–Concept associations to know that the writer means the Apple Corporation and not the fruit when they just say “apple”.

Our generative model makes the simplifying assumption that it is the Concepts themselves that generate terms in a text. When a writer wishes to write a document or formulate a query about the Apple Corporation, we say that the Concept of *Apple_Corporation* is actually generating the terms in the text directly. Therefore, a query about *Apple_Corporation* is likely to contain terms like “computer” and “iPhone” due to the Concept *Apple_Corporation*'s propensity to generate such terms.

To be more precise, our generative model states that texts are generated term by term from some topic Concept, where a Concept is an object encapsulating a general idea or topic that can result in the generation of text. The a priori probability of a given Concept C being the topic Concept of our text is denoted $P(C)$. For every term t, there is a probability $P(t|C)$ that C will generate t as the next term in the text. Documents, queries, and other forms of text are all considered to be of one type, TermSequence. By incorporating knowledge of all possible Concepts and their probability of generating each term, it is possible to take a TermSequence and work backwards to find the Concepts that generated it. In order to get this knowledge, we extract a set of fundamental AtomicConcepts from Wikipedia.

4.1. AtomicConcept

An AtomicConcept is the simplest type of Concept. Like all Concepts, an AtomicConcept has an a priori probability of being a topic Concept (denoted $P(A)$ by convention when the Concept is atomic), and probabilities $P(t|A)$ of generating term t. We view each article in Wikipedia as a long TermSequence that was generated by some AtomicConcept. Since every article is unique, there is a one to one mapping between articles and the AtomicConcepts that generate them.

In order to estimate the a priori probability $P(A)$, we look at the relative number of inter-article links that point to A's article.

$$P(A) = \frac{number\ of\ incoming\ links}{number\ of\ links\ in\ Wikipedia}$$

Since Wikipedia articles link to the other articles they discuss, the fraction of incoming links is a good estimate of the likelihood that an article's subject (its AtomicConcept) will be talked about.

To estimate $P(t|A)$, we view the article body text as a sample of terms generated by *A*. The probability of *A* generating a term *t* is:

$$P(t|A) = \frac{count(t, A)}{number\ of\ words\ in\ A} \quad (1)$$

Because articles have finite length, some terms relevant to *A* won't actually show up in the body text of the article. For each term not present in the article, we smooth the distribution by estimating the probability of *A* generating t to be the probability of t occurring in the English language.[2]

[2] In Wikimantic, we use Microsoft n-Grams to give us $P(t)$. Because probabilities from Wikipedia and Microsoft n-Grams each sum to 1, the sum of $P(t|A)$ over all *t* then equals 2. In practice, estimated probability values for AtomicConcepts are always stored as elements of normalized collections, which ensures that no probability value falls outside the range [0,1].

4.2. MixtureConcept

Although Wikipedia covers a wide range of topics, it would be overly simplistic to assume that each and every real world text can be accurately summarized by just one AtomicConcept. A query about Apple Inc.'s profits on the iPod Touch might better be summarized with a mixture of the AtomicConcepts *Apple_Inc.*, *iPod_Touch*, and *Profit_(accounting)*. Thus we instead model the topic using a MixtureConcept which is a set of weighted AtomicConcepts. When a MixtureConcept generates a term, it randomly selects one of its AtomicConcepts to generate in its stead. The weight of an AtomicConcept tells us the probability that it will be the one selected to generate, and all weights necessarily sum to 1. Like all Concepts, a MixtureConcept M has an a priori probability $P(M)$ of being the topic, and probabilities $P(t|M)$ of generating a given term t.

$$\begin{aligned}&\text{Let MixtureConcept } M = \{(w_i, A_i) | i = 1 \ldots n\}\\ &\quad \text{where } w_i = \text{the weight of } A_i \text{ in } M\\ &P(M) = \sum_{i=1}^{n} w_i * P(A_i)\\ &P(t|M) = \sum_{i=1}^{n} w_i * P(t|A_i).\end{aligned}$$

If a term sequence s discusses Apple Inc., summarizes its profits and briefly mentions the iPod Touch, the MixtureConcept for s may look something like:

$$M_s = \{(0.5, \textit{Apple\_Inc.}), (0.3, \textit{Profit\_(accounting)}), (0.2, \textit{iPod\_Touch})\}$$
$$P(M_s) = 0.5 * P(\textit{Apple\_Inc.}) + 0.3 * P(\textit{Profit\_(accounting)}) + 0.2 * P(\textit{iPod\_Touch}).$$

To find the likelihood of the term "iPhone" in any term sequence with M_s as its topic, one would calculate

$$\begin{aligned}P(\text{“}\textit{iPhone}\text{”}|M_s) = {} & 0.5 * P(\text{“}\textit{iPhone}\text{”}|\textit{Apple\_Inc.})\\ & + 0.3 * P(\text{“}\textit{iPhone}\text{”}|\textit{Profit\_(accounting)})\\ & + 0.2 * P(\text{“}\textit{iPhone}\text{”}|\textit{iPod\_Touch}).\end{aligned}$$

The key problem is how to estimate the weight w_i of each AtomicConcept. In the following, we first present a method that uses the content of a concept's article to estimate w_i and then a second method that uses references between concepts.

4.3. Content-based topic modeling

To build a MixtureConcept M that represents the meaning of a TermSequence s constituting a query, we first populate the set with AtomicConcepts and then weight the AtomicConcepts. Recall that there is a 1–1 correspondence between AtomicConcepts and Wikipedia articles; thus an AtomicConcept is essentially a Wikipedia article. Our base method uses the content of a concept's article to estimate the concept's weight in the MixtureConcept.

To construct the elements of the set of AtomicConcepts comprising the MixtureConcept, we look at every subsequence of terms in s and attempt a direct lookup in Wikipedia. Any article that has a title that matches a subsequence of terms in s is added to the set. Any article that is disambiguated by a page whose title matches a subsequence of terms in s is also added to the set. Finally, all articles that share a disambiguation page with an article already in the set are added.[3] For example, if s = "Steve Jobs resigns":

Steve → Matches title of disambiguation page *Steve*. Add all articles disambiguated by that page.
Jobs → Matches the title of a redirect page that points to *Jobs_(Role)*. Add *Jobs_(Role)* and all other articles that *Jobs_(disambiguation)* link to.
Resigns → Matches the title of a redirect page that points to *Resignation*. Add *Resignation* and all other articles that *Resignation_(disambiguation)* link to.
Steve Jobs → Matches the title of the article *Steve_Jobs*.
Jobs Resigns → Matches nothing, so no articles added.
Steve Jobs Resigns → Matches nothing, so no articles added.

[3] In most cases we get the correct Wikipedia article added to our MixtureConcept; thus we have not seen the need to use anchor text from Wikipedia and leave that to future work.

Once our unweighted set M is populated, it will contain a large number of candidate AtomicConcepts of varying degrees of relevance, and we rely on weights to mitigate the impact of spurious concepts. We weight each AtomicConcept A_i according to the probability that every term in s was generated by that AtomicConcept, ignoring stop words.

$$w_i = P(A_i|s) = \prod_{j=1}^{|s|} P\left(A_i|t_j\right) \tag{2}$$

$$P\left(A_i|t_j\right) = \frac{P\left(t_j|A_i\right) * P(A_i)}{P\left(t_j\right)} \tag{3}$$

This weighting schema ensures that an AtomicConcept will only get a high score if it is likely to generate all terms in the sequence. A Concept like *Jobs_(Role)* may have a high probability of generating "jobs", but its low probability of generating "steve" will penalize it significantly. We can expect the Concept *Steve_Jobs* to generate "steve", "jobs", and "resigns" relatively often, which would give it a much larger weight than *Jobs_(Role)* would get.

4.4. ReferenceRank topic modeling: M_R

Our second method, ReferenceRank, uses references between AtomicConcepts to estimate the weights w_i. In M_R, AtomicConcepts are weighted according to the probability that they describe the sense of a given term in s, rather than the probability that they will generate a given term in s.

Consider the following example where M might be misleading if it is weighted by probability of generating.

$$M1 = \{(0.5, \text{Apple\_Inc.}), (0.5, \text{Whole\_Foods})\}$$
$$M2 = \{(0.5, \text{Apple\_Inc.}), (0.2, \text{iPhone}), (0.2, \text{Apple\_Safari}), (0.1, \text{iPad})\}$$

In the text described by $M1$, the topic is 50% about Apple Inc. and 50% about the grocery store Whole Foods. In the document described by $M2$, the topic is 50% about Apple Inc. and 50% about various Apple products. Since *iPhone*, *Apple_Safari*, and *iPad* are all concepts that are likely to generate the term "apple" (referring to the company), one would expect Apple Inc. to be referenced more often in $M2$ than $M1$. However, Apple Inc. is weighted equally in $M1$ and $M2$. It's subtle, but there is a very real difference between the probability that a Concept will generate terms in our query and the probability that a Concept will be referred to by concepts associated with other terms in our query. To account for this, we extend our generative model by making the claim that Concepts generate references to other concepts as well as terms.

Given that AtomicConcept A_1 generated a reference to another concept, the probability that the referenced concept is A_2 is estimated as the probability that clicking a random link in A_1's article will lead directly to the article for A_2.

$$P\left(R_{A_2}|A_1\right) = \frac{\textit{number of links from } A_1 \textit{ to } A_2}{\textit{total number of links originating at } A_1}$$

The probability that a MixtureConcept M will generate a reference to a Concept C (denoted R_C) is just a mixture of the probabilities of the AtomicConcepts in M generating a reference to C:

$$P(R_C|M) = \sum_{i=1}^{n} w_i * P(R_C|A_i),$$

where $w_i =$ the weight of A_i in M (Eq. (2))

For a TermSequence s, we compute the special MixtureConcept M_R that contains all relevant AtomicConcepts weighted by their probability of being referenced.

$$\text{Let MixtureConcept } M_R = \{(w_i, A_i)|i = 1...n\}$$
$$\text{where } w_i = P\left(R_{A_i}|M\right)$$

This reweighting in M_R is very similar to one iteration of the PageRank algorithm [30], where nodes in a graph vote for other nodes to which they link. In our case, AtomicConcepts in M vote for other AtomicConcepts in M, and the power of each vote is proportional to the weighting of that AtomicConcept in M.

5. Disambiguation

Once the topic model (M or M_R) has been populated and weighted, it can be used to guide disambiguation. Each term t in a query is associated with one or more weighted AtomicConcepts in the topic model; thus in the simplest case, disambiguation of a

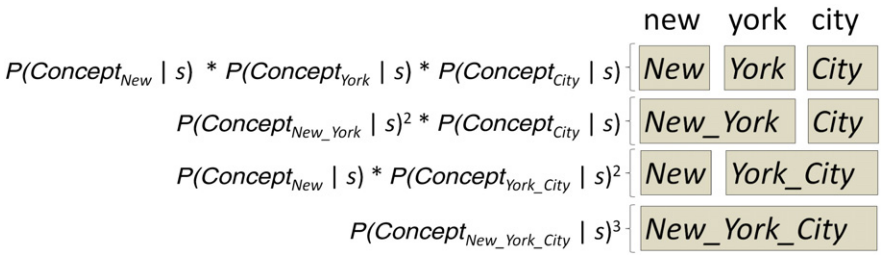

Fig. 1. Product method's scoring of four possible disambiguations of "new york city".

term t is a matter of selecting the AtomicConcept associated with the *t* that has the highest weight. In the case of topic model *M*, this makes the somewhat strong assumption that the probability of an AtomicConcept generating the string *s* is roughly equal to its probability of being the correct mapping for a term in *s*. In the case of topic model M_R, the weight of an AtomicConcept has been adjusted to reflect its probability of describing the sense of a term in s.

However, reliance on the topic model M_R by itself is problematic due to Wikipedia's relatively sparse link structure. "Do Life Savers cause tooth decay?" is a perfectly reasonable query, but there are no direct links between the articles for Life Savers and tooth decay. This means that neither will receive a vote and therefore their weights in M_R will be zero. Although M_R is useful when links are found, it must be supplemented with information from *M*. For this reason, the mixture $P(A|s) = (1 - d) * M + d * (M_R)$ is used. The optimal value of d is determined experimentally.

Another issue is the need to handle sequences of words that should be treated as a single concept. In many disambiguation papers [4,11,25,31], the important term strings are assumed to be marked ahead of time and the system must simply choose the single best Wikipedia article for the marked string. For queries, the number of mappings are not known a priori, which makes disambiguation considerably more difficult. Does "Life Saver" refer to the brand of candy or a person who saved a life? If we are talking about junk food, then "life saver" should entail a single mapping to the AtomicConcept *Life_Saver*, otherwise it entails two separate mappings to *Life* and *Saver*. Although these kinds of conflicts seem like they should be rare, the vast coverage of Wikipedia actually makes them common. Company names, book titles, and music album titles are particularly troublesome since they are often common phrases; moreover, they are often the topics of graphs in popular media and thus occur in user queries for these graphs.

Thus we need a more robust method of disambiguation. Our approach is to consider every possible sequence of words as a term for disambiguation. Thus if a problematic sequence of terms like "life saver" or "new york city" is found, every possible breakdown of the sequence is disambiguated. Each breakdown yields a unique candidate set of disambiguations that is scored according to its probability of being the correct one. The scoring is calculated using either the product method or the averaging method, as described in the following subsections.

5.1. Product method

The product method scores a candidate set as the product of the probabilities of each mapping of term to AtomicConcept. Fig. 1 depicts the four candidate sets that are considered when the string "new york city" is broken down.[4] We use italics to refer to AtomicConcepts by name, so $P(Concept_{New}|s)$ refers to the probability of the Concept *New* being the disambiguation of the term "new".

The first set contains the three AtomicConcepts *New*, *York*, and *City*. The score of the set is simply the product of their probabilities multiplied together. When n adjacent terms should be disambiguated as a single entity, the Product method scores it as *n* disambiguations of the entity,[5] as shown by the fourth row in Fig. 1, where the score for the sequence "new york city" is $P(Concept_{New\_York\_City}|s)$ to the third power.

5.2. Averaging method

The Averaging method treats a set of possible disambiguations as an average of the probabilities of each mapping of term to AtomicConcept. Under the Averaging method, a set's score is simply equal to the average of the probability values of all Atomic Concepts in the set; once again, *n* adjacent terms that were disambiguated as referring to a single entity are counted as *n* disambiguations. For example, the fourth line of Fig. 2 shows the sequence "new york city" being disambiguated as a single entity but

[4] Although this example is a named entity and much research has focused on named entity recognition (see Section 2), the example is a clear illustration of our scoring method. Other examples of multiple word terms that are disambiguated are shown in Table 1.

[5] Otherwise, longer terms would always be preferred over several shorter ones.

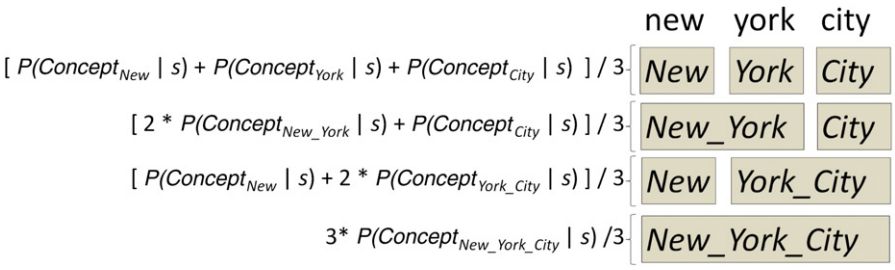

Fig. 2. Mixture method's scoring of four possible disambiguations of "new york city".

the score in this case is just the probability of the concept *New_York_City* (i.e., the average of the scores for three disambiguations of the sequence).

5.3. Evaluation

Our system, named Wikimantic, includes four alternative methods for disambiguation: the product and averaging methods with MixtureConcept M as the topic model for the term sequence s and the product and averaging methods with $(1-d) * M + d * M_R$ as the topic model. Each method was evaluated using 70 queries from the Trec 2007 QA track and 26 queries collected for our information graphic retrieval project. The QA track was chosen because we intend to eventually incorporate Wikimantic into a larger system that operates on short grammatically correct full sentence questions, but it is worth noting that Wikimantic is in fact entirely agnostic to the grammatical structure of its input. The queries acquired from the Information Graphic Retrieval Project were collected from human subjects who were given information graphics and told to write queries they might have used to find them. All queries contain at least one (but usually more) salient word that must be disambiguated. The word count of each query is no less than 4 and no greater than 15. Out of the 850 words in the set, evaluators identified about 349 nouns (they disagreed on a couple due to ambiguous phrasing of the queries). About 110 words were content words that were not nouns. We present results for disambiguating just nouns and for disambiguating all non-function words.

Table 1 displays some short queries and a few interesting terms disambiguated by Wikimantic, along with the Wikipedia concepts that they were equated with. Of particular interest are terms such as *airport code*, *comic strip*, *hybrid vehicles*, *endangered animals*, and *price of oil* which are multi-word terms that are not named entities. For example, *airport code* is correctly disambiguated by Wikimantic as a single concept but which could have been erroneously interpreted as two separate concepts with multiple possible interpretations for *code*.

To measure correctness, we gave the system results to two evaluators and instructed them to decide for each term whether the linked page correctly described the meaning of the word as it was used in the query. The general rule was that a disambiguation was wrong if a better page could be found for the term. For non-nouns, it was considered correct if a verb or adjective was linked to its noun-equivalent article. For example, it would be acceptable to annotate the term "defect" (to betray) with the page "Defection". If a term appeared in the query with a sense that has no equivalent article in Wikipedia, the evaluators were instructed to mercilessly mark the output wrong. The Kappa statistic [9] measures the degree of inter-rater agreement between evaluators while discounting for the likelihood of chance agreement. The Kappa statistic for the two evaluators was .77 which represents substantial agreement.

Tables 2 and 3 present statistics on precision and recall for the four methods. Precision is equal to the number of terms correctly mapped to concepts divided by the number of terms mapped to concepts by the system. Recall is equal to the number of correct mappings divided by the number of terms fed to the system. As a baseline, we input each term to WordNet and, using the correct part of speech, selected the interpretation with the highest frequency. This produced baseline F-measures of 58.29 for nouns and 59.25 overall.

5.4. Discussion

All of the methods had F-measures that exceed the baseline. Overall, the product method fared better than the averaging method, and performance was better on nouns than on non-nouns. With the averaging method, it is possible for an obviously incorrect mapping to be offset by a high scoring one. With the product method, a mapping with a near-zero probability will cause the score for the entire set to be near-zero. The product method is therefore a more conservative scoring method that favors well rounded sets over sets with some likely and some unlikely references. The exceptional performance on nouns seems to be partly due to Wikipedia's greater coverage of nouns. Additionally, Wikimantic did not incorporate a stemmer, which occasionally prevented it from recognizing matches between alternate conjugations of the same verb.

For each of the two methods described in Section 5.1, we evaluated Wikimantic using the combination $(1-d) * M + d * M_R$ with varying values of d. Improved performance occurred for small values of d ($d < .2$). Although the optimal value of d was found to be very small ($d = 0.0001$), the effects of ReferenceRank were still surprisingly significant. MixtureConcepts often get

Table 1
Example Wikimantic disambiguation results.

Query: What is the 3-character airport code for Dulles Airport?	
Term	Concept identified by Wikimantic
airport code	International_Air_Transport_Association_airport_code
Dulles Airport	Washington_Dulles_International_Airport
Query: Who was the creator of the comic strip Blondie?	
Term	Concept identified by Wikimantic
creator	Creativity
comic strip	Comic_strip
Blondie	Blondie_(1968_TV_series)
Query: How many new hybrid vehicles were sold in the United States in 2004?	
Term	Concept identified by Wikimantic
hybrid vehicles	Hybrid_electric_vehicle
Query: How has the price of oil change during the last few months?	
Term	Concept identified by Wikimantic
price of oil	Price_of_petroleum
Query: At what rate has the amount of endangered animals increased?	
Term	Concept identified by Wikimantic
endangered animals	Endangered_species
Query: How reliant are average citizens on communication towers to connect to the internet, etc.?	
Term	Concept identified by Wikimantic
communication	Telecommunication
towers	Transmission_tower
Query: On what date did Harriet Miers withdraw her nomination?	
Term	Concept identified by Wikimantic
withdraw	Withdrawal
Query: Who invented the Rubik's Cube?	
Term	Concept identified by Wikimantic
invented	Invention
Query: How many years did it take to build St Peter's Basilica?	
Term	Concept identified by Wikimantic
build	Construction

Table 2
Performance on nouns only.

	Topic model: M		Model: $(1-d) * M + d * M_R$	
	Averaging	Product	Averaging	Product
Precision	78.71	80.51	81.38	82.76
Recall	75.21	76.93	77.65	79.08
F-measure	76.92	78.68	79.47	80.88

Table 3
Performance on all non-function words.

	Topic model: M		Model: $(1-d) * M + d * M_R$	
	Averaging	Product	Averaging	Product
Precision	66.82	68.28	69.52	70.57
Recall	61.47	62.45	63.96	64.61
F-measure	64.04	65.23	66.62	67.46

weighted in such a way that one AtomicConcept has virtually all the weight, which gives it extremely high voting power. The top AtomicConcept's votes are then so powerful that they have disproportionate sway over the lesser AtomicConcepts. For example, during ReferenceRank, if a Concept $C1$ in M with a weight of 0.995 votes for a Concept $C2$ in M with a weight of 0.0003, $C2$ could get a huge boost because $C1$'s voting power is proportional to its huge weight. The d value works best when small because it stops the ReferenceRank votes from completely overriding the initial scores from M.

Our results show that our system Wikimantic has very good success at disambiguating terms in short queries, even without capitalization or a priori identification of multi-word strings that should be mapped to a single concept.

6. Extending Wikimantic to query expansion

This section looks into how to use Wikimantic results to improve retrieval performance. Since Wikimantic makes it possible to disambiguate query terms and associate them with Wikipedia concepts, a natural way of using Wikimantic results is to expand original queries with terms from the identified Wikipedia concepts. These expansion terms are expected to improve the retrieval of relevant documents since they could bridge the vocabulary gap between original queries and relevant documents. For example, Wikimantic would map the query "superbowl commercials" to the Wikipedia concept "Super_Bowl_advertising". Expanding the query with terms from this concept, such as "advertising", would enable the retrieval of more relevant documents such as the ones mentioning "superbowl" and "advertising" but not "commercials", which would otherwise have a low ranking without query expansion.

In this paper, we focus on the problem of microblog retrieval, whose goal is to retrieve relevant tweets for a given topic [29]. As noted in Section 1, this search domain is chosen for several reasons, most notably that tweets are much shorter than traditional documents and are thus similar in length to the limited text (caption, axis labels, and intended message) that will be available for a system designed to retrieve information graphics. We first describe a commonly used baseline method for microblog retrieval, and then discuss three methods that use Wikimantic results to further improve the performance. In particular, the first two methods use Wikimantic results for query expansion while the last one uses it to adjust the weight of query terms.

6.1. Baseline method for microblog retrieval

Microblog retrieval is similar to ad hoc search on Twitter's site, which aims to find relevant tweets for a given topic at a specific time. Thus, the main difference from the traditional ad hoc retrieval task is that the query contains not only a topic but also a timestamp, and only relevant tweets that were posted before the timestamp of the query should be retrieved.

Tweets are much shorter than traditional documents since a tweet can contain at most 140 characters. Such a unique property makes it necessary to examine whether the methods used for traditional ad hoc retrieval are still optimal for this new search task. There are three commonly used retrieval heuristics, i.e., term frequency (TF), inverse document frequency (IDF) and document length normalization [12]. However, not all of them are appropriate for microblog retrieval. Specifically, the length limitation of a tweet often leads to similar short lengths and low term frequency (0 or a small integer, often 1, for a specific term), which makes it inappropriate to use TF and document length frequency heuristics. Consequently, it appears that considering just IDF weighting [17] is an appropriate heuristic for retrieval of relevant tweets. Thus, the baseline retrieval method used in this paper is the following:

$$S(Q,D) = \sum_{t \in Q \cap D} ln \frac{N+1}{\mathrm{df}(t)}, \tag{4}$$

where Q denotes a query, D denotes a tweet, t denotes a term, N is the number of tweets that were posted before the query time in the collection, $\mathrm{df}(t)$ is the number of tweets that were posted before the query time and contain term t, and $S(Q,D)$ is the relevance score of the tweet D for the given query Q. Thus tweets D with large values of $S(Q,D)$ include more terms from the query that are uncommon in other tweets, thereby making these tweets more likely to be relevant to the query Q. This baseline method is referred to as **BL**.

Note that we have conducted preliminary experiments using the collection from Microblog Track 2011 [29], and found that the retrieval function shown in Eq. (4) is more effective than Okapi [34], which is one of the state of the art retrieval functions (e.g., 0.4497 versus 0.3932 measured using Precision@30). This observation has also been confirmed by other researchers [41,14].

The next sections describe our efforts in leveraging Wikimantic results to improve retrieval performance.

6.2. Query expansion via Wikimantic

Query expansion is a commonly used strategy in IR to improve retrieval performance since it is effective in bridging the vocabulary gap between terms in a query and those in the documents. The basic idea is to expand the original query with terms that could potentially match relevant documents that use terms other than those explicitly given in the original query. For example, given the query "car costs", if we could expand the query with related terms such as "automobile" and "expenditures", we would be able to retrieve relevant documents mentioning "automobile" instead of "car" or "expenditures" instead of "costs". The challenge is how to select expansion terms and appropriately assign weights to them so that they exert the appropriate influence in evaluating candidate documents for retrieval.

Recall that the proposed Wikimantic system can construct a MixtureConcept that represents a topic and that the MixtureConcept is a set of weighted AtomicConcepts. A natural way of using Wikimantic results for query expansion is to utilize the information from the MixtureConcept to identify terms that are highly relevant to those in the original query.

Formally, given a query Q, we first use Wikimantic to construct the MixtureConcept $M(Q) = \{(w_i, A_i)|i = 1...n\}$, which is a set of AtomicConcepts (Wikipedia articles) and their associated weights. We can then use a term weighting method to select expansion terms from these Wikipedia articles and expand the original query with the weighted expansion terms.

In particular, we propose the following term weighting method to select expansion terms t.

$$ExpWeight(t|M(Q)) = \sum_{A_i \in M(Q)} P(t|A_i) \times w_i \times ln\frac{N+1}{\mathrm{df}(t)}, \tag{5}$$

where $P(t|A_i)$ is the likelihood of generating term t from the AtomicConcept A_i (Eq. (1) in Section 4.1) and w_i is the weight of AtomicConcept A_i in $M(Q)$. N and df(t) are again respectively the number of tweets and the number of tweets containing term t. $ln\frac{N+1}{\mathrm{df}(t)}$ represents the importance of a term and is computed using the IDF weighting [17].

Thus the above term weighting for query expansion captures three factors: (1) the likelihood that the term t can represent the AtomicConcept A_i; (2) the likelihood that AtomicConcept A_i is relevant to the original query; and (3) the importance of term t in retrieving relevant documents.

We can then select K terms for query expansion based on the weighting method described in Eq. (5), and the weighting of the newly added expansion terms would be controlled by a parameter α. We denote $ExpT(M(Q))$ as the selected K expansion terms for query Q. Thus, the retrieval function based on query expansion can be written as:

$$S(Q,D) = \sum_{t \in Q \cap D} ln\frac{N+1}{\mathrm{df}(t)} + \sum_{t \in ExpT(M(Q)) \cap D} \alpha \times ExpWeight(t|M(Q)), \tag{6}$$

where α controls how much we trust the expansion terms. When $\alpha = 0$, we use only original query terms. When its value gets larger, we put more trust on the expansion terms.

Note that $M(Q)$ is the constructed MixtureConcept for query Q. In our experiments, we consider two possible values for $M(Q)$: (1) the original MixtureConcept, which is constructed using the methods described in Section 4; and (2) the subset of AtomicConcepts from the original MixtureConcept that disambiguates the terms in the query Q, which is constructed using the method described in Section 5. Notice that the difference between the two variations for $M(Q)$ is that the first is a set of AtomicConcepts where no possible disambiguation of a term in the query Q has been discarded whereas the second includes only the AtomicConcepts that have been selected as the disambiguation of a term in the query Q. The expansion methods corresponding to these two scenarios are referred to as **QE-ALL** and **QE-DIS** respectively.

6.3. Concept-based query term weighting

Our third strategy uses Wikimantic results to adjust the term weighting based on the relations among query terms. Previous studies [42,39] have shown that concept-based term weighting regularization is effective at improving retrieval performance, but they mainly focus on utilizing the co-occurrence among query terms to detect query concepts, which could lead to inaccurate results. Fortunately, Wikimantic disambiguation results enable us to identify query concepts based on Wikipedia articles, and we now explore how to use these disambiguated concepts to adjust the term weighting.

The idea of concept-based term weighting is motivated by the limitation that traditional retrieval models ignore the relations among query terms and may favor documents that are relevant to a subset of query concepts. For example, given a query "BBC staff cut", traditional retrieval models may incorrectly assign a higher relevance score to documents covering "BBC staff" which is a single concept in the query than to those covering "BBC" and "cut" which are two individual concepts. It appears more desirable to favor documents covering more query concepts.

Thus, given a query, we first apply the methods described in previous sections to disambiguate query terms through Wikimantic concepts. As a result, every query term is then associated with an AtomicConcept, and multiple terms may correspond to the same AtomicConcept. We then assume all the terms corresponding to a single AtomicConcept forms a query concept. We now explain how to use the query concepts information to adjust the term weighting.

As before, we select the K best terms for query expansion based on their weights $ExpWeight(t|M(Q))$ given by Eq. (5). Motivated by the idea of search result diversification, we propose to discount the weight of a term in a document based on the number of more important terms from the same concept that have been covered in the same document as follows:

$$CWeight(t,D) = \frac{TermWeight(t)}{Rank(t, QC(t), D)}, \tag{7}$$

where $CWeight(t)$ is the adjusted weight of term t and $TermWeight(t)$ is the weighting of term t and is computed using either IDF weighting [17] for query terms or the weighting shown in Eq. (5) for expansion terms. $QC(t)$ denotes the query concept that contains term t, and $Rank(t,QC(t),D)$ denotes the rank of term t when we sort all the query terms and expansion terms from $QC(t)$ that occur in document D based on their $TermWeight$ values in a decreasing order. Thus the retrieval function based on concept weighting can be written as:

$$S(Q,D) = \sum_{t \in Q' \cap D} ln\frac{N+1}{\mathrm{df}(t)} + \sum_{t \in Q_c \cap D} \frac{ln\frac{N+1}{\mathrm{df}(t)}}{Rank(t, QC(t), D)} + \sum_{t \in ExpT(M(Q)) \cap D} \frac{\alpha \cdot ExpWeight(t|M(Q))}{Rank(t, QC(t), D)}, \tag{8}$$

where Q' denotes a set of query terms that are not associated with any AtomicConcepts, Q_c denotes the terms in the query that are associated with an AtomicConcept, and $ExpT(M(Q))$ denotes the K selected expansion terms for query Q. The basic idea is to favor documents covering all the query concepts through reducing the weights of query terms whose corresponding concepts have been covered by more important terms. The method is referred to as CW. Once again, α controls how much emphasis is placed on the expansion terms versus the terms in the original query.

6.4. Experiments

We conducted experiments over two TREC collections to examine whether using Wikimantic results can improve retrieval performance.

6.4.1. Experiment design

We use two standard TREC collections in the experiments: (1) *Microblog11*: Tweets11 corpus [29], 50 queries and the corresponding relevance judgments; and (2) *Microblog12*: Tweets12 corpus, 60 queries and the corresponding relevance judgments. The Tweets11 corpus contains around 10M tweets spread over 2 weeks, i.e., Jan. 24, 2011 to Feb. 8, 2011. Both query sets and judgments were created by NIST assessors. The judgments were made using a three-point scale: "Not relevant", "Minimally relevant" and "Highly relevant". The results are evaluated with multiple representative measures including MAP (mean average precision) and P@30 (Precision at top 30 documents). All measures are computed with both "minimally relevant" and "highly relevant" as the required level of relevance.

6.4.2. Effectiveness of the proposed methods

We evaluated the effectiveness of the proposed methods and compared them with two baseline methods: (1) **BL**, which is the basic retrieval method without query expansion as described in Section 6.1; and (2) **QE-WN**, which is a standard WordNet query expansion method that expands a query with synonyms of query terms. The performance results are summarized in Table 4.

It is clear that our proposed query expansion methods are effective at improving retrieval performance, and the improvement is consistent over both collections and measures. On the contrary, the baseline query expansion method **QE-WN** fails to improve the performance.

Moreover, the concept weighting strategy **CW** is less effective than methods **QE-DIS** and **QE-ALL** in most cases. In fact, the best performance of our proposed methods, **QE-DIS** can be ranked as the top 1 run in the microblog 2011 track [29] (P@30 for the best reported run is 0.4551, and that for the second best is 0.4401.). Although Wikipedia was used by some participants at the track [36], their reported performance (0.3014 measured with P@30) is much worse than ours.

Since the effectiveness of the proposed query expansion methods is closely related to Wikimantic's disambiguation accuracy, the good performance of our expansion methods also indicates that the disambiguation results are satisfactory. For instance, Table 5 shows a few example queries and their disambiguation results. Of particular interest are terms such as *somalian piracy* and *superbowl commercials* which are multi-word terms that are not named entities. In addition, the query *assange nobel peace* contains two partial names (*assange* and *nobel peace*) which are correctly identified as Julian_Assange and Nobel_Peace_Prize.

Table 4
Performance comparison over the two TREC data sets.

	Microblog11		Microblog12	
Methods	P@30	MAP	P@30	MAP
BL	0.4490	0.4085	0.3141	0.2127
QE-WN	0.3973	0.3697	0.2949	0.2027
QE-ALL	0.4558	0.4235	0.3633	0.2438
QE-DIS	0.4585	0.4266	0.3650	0.2477
CW	0.4544	0.4142	0.3661	0.2428

Table 5
Example Wikimantic results for Microblog Track queries.

Query	Concepts identified by Wikimantic
black swan natalie portman	Black_Swan_(film) Natalie_Portman
steve jobs health	Steve_Jobs Health
somalian piracy	Piracy_in_Somalie
superbowl commercials	Super_Bowl_advertising
king's speech award	The_King's_Speech Award
assange nobel peace	Julian_Assange Nobel_Peace_Prize
giffords recovery	Gabrielle_Giffords Cure

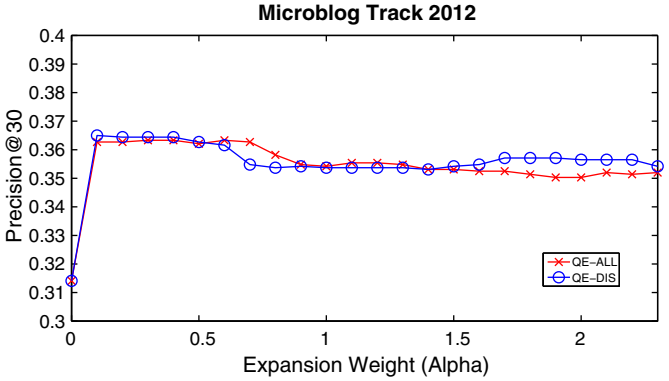

Fig. 3. Performance sensitivity w.r.t. α.

6.4.3. Parameter sensitivity

Our best query expansion methods have two parameters: (1) α, i.e., the regularization parameter for the weight of expansion terms in Eq. (6); and (2) K, i.e., the number of expansion terms added to each query. Fig. 3 shows the performance sensitivity curve with respect to α, and Fig. 4 shows the curve with respect to K. Both figures are computed using *Microblog12* data set, and the trends are similar on the other data set. Fig. 4 shows that performance improves substantially as expansion terms are added and then levels off. Recall that expansion terms are weighted (Eq. (5)) and as more expansion terms are added, they have lower weights and thus a lesser impact on the computation of the relevance of the document to the query (Eq. (6)). Also note that although non-zero values of α improve performance, precision does not change markedly for the different values of α shown in Fig. 3.[6] Thus we see that performance is robust with respect to both parameters.

6.4.4. Discussions

To better understand the performance of the proposed methods, we further analyze their performance and make the following interesting observations.

First, the disambiguation results of Wikimantic enable better query understanding. For example, consider the query "The Daily". Both query terms are common words, so traditional keyword matching methods would return many non-relevant documents matching either or both of the terms. In fact, the P@30 of the **BL** for this query is 0. Fortunately, with the help of Wikimantic, we are able to connect query terms with Wikipedia concepts such as "The Daily News Corporation" and "The Daily Newspaper", which indicate the real information need behind the query. The expansion methods using related terms from these concepts can improve the performance to 0.87 measured by P@30. Similar observations can be made for other queries with common words, such as "release of The Rite".

Second, the expansion terms from the constructed MixtureConcept are effective in bridging vocabulary gaps, especially for those with named entities. For example, for the query "Hugo Chavez", our system can find related terms such as "Venezuela" and

[6] Note that even when α is greater than 1, the expansion terms have less impact than the terms explicitly appearing in the query due to the way in which their contribution is computed in Eq. 6.

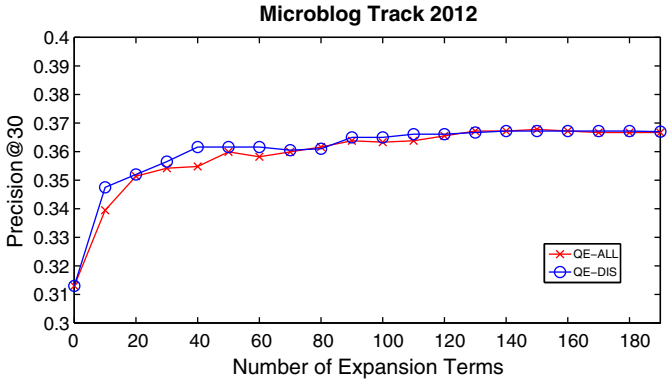

Fig. 4. Performance sensitivity w.r.t. the number of expansion terms.

"leader". Similar queries are "Steve Job's health", "Oprah Winfrey half-sister", and "Michelle Obama", where sequences of terms should be grouped together and associated with a single concept from which related terms are identified.

Third, query expansion may hurt performance for queries when the concepts contributing expansion terms are not the only important ones in the query. For example, the expansion terms for the query "Facebook privacy" mainly come from the concept "Facebook", but both "privacy" and "Facebook" are important concepts. As a result, the retrieval result based on expansion terms may over-favor tweets mentioning Facebook instead of the privacy concern. Other similar queries are "Starbucks Trenta cup" and "Chicago Blizzard".

Fourth, although the overall performances of **QE-ALL** and **QE-DIS** are similar, their per-query performance could be different. For the query "NCIS", **QE-DIS** performs better because the disambiguation results can effectively identify that the correct concept is "NCIS TV series" as opposed to other concepts such as "National Criminal Intelligence Service" or "New Century Infusion Solution". However, for the query "Saleh Yemen overthrow", **QE-ALL** performs better because the disambiguated concept "Yemen" is only part of the query and cannot cover information about "Saleh" and "overthrow".

Finally, the concept weighting method **CW** seems to be less effective than **QI-ALL** and **QE-DIS**. One of the reasons is that the method can help queries with multiple-term concepts but does not help queries with single term concepts. Consider the query "illegal immigrant laws" as an example; the baseline method would return many non-relevant documents covering "illegal immigrant", but the proposed concept weighting method **CW** would improve the performance by returning relevant documents covering both concepts, i.e., "illegal immigrant" and "law". A few other example queries where the concept weighting method improves retrieval are "British Government cut", "horse race betting" and "Keith Olermann new job". However, the concept weighting method won't help queries with single-term concepts, such as "Yemen Saleh overthrow".

In summary, we have proposed effective query expansion methods that use Wikimantic results to improve retrieval performance. Specifically, query expansion methods are more effective, but they may introduce additional computational costs compared with the baseline method. Concept-based query term weighting is less effective in terms of the overall performance, but it can improve the retrieval performance for a subset of queries and the computational cost is as cheap as the baseline method.

7. Optimization of Wikimantic

Since Wikimantic is intended to be used as part of a real-time information retrieval system that responds to user queries, it is important that the implementation be fast. Wikimantic must retrieve a large amount of data from the node graph for each query, so most of our optimizations involve avoiding unnecessary disk accesses and being selective with the data we add to our node graph during compilation. We found that keeping the node graph on a solid state disk worked reasonably well since the size of the node graph is small and read speeds are important.

One time consuming stage of the compilation process is extracting term frequencies from each article. In particular, very long articles take more time to tokenize and tend to have a larger number of unique tokens which must be recorded. To mitigate these costs, we simply cut off each article after the 30,000th character. Since Wikimantic rests on the assumption that each article is focused on a single topic, 30,000 characters worth of text should be a large enough sample to determine the most salient words. If a word does not appear often in the first 30,000 characters, we generally assume it is not critically relevant to the article.

The amount of space an extracted article takes up on disk is proportional to the number of unique words in the article. This means that the least frequent (and therefore, least relevant by Wikimantic's metric) word in the article will take up the same amount of space on disk as the most frequent word. We found that ignoring infrequent terms is an effective way of speeding up compilation and limiting the size of our node graph. Specifically, we chose a threshold $t = 0.1$ and ignored each term in the article that comprised less than $t\%$ of the total terms. Ignoring unimportant terms has the additional very important advantage of speeding up disambiguation, since insignificant entries will not be read from the disk later.

The most time consuming stage of the compilation process is extracting the inter-article links from the Wikipedia dump files and creating corresponding edges in our node graph. Each time an edge is created in our node graph, it is necessary to load both the starting node and the end node into memory so that they can be properly registered with the edge. This is a bottleneck, so being able to reduce the number of reads can help significantly. Fortunately, the dump files seem to be structured in such a way that link entries from the same node appear close to each other in the file. Using an LRU cache, we are able to keep the top 1000 or so recently used nodes in RAM, which lets us quickly create links between nodes we have recently used.

After compilation, Wikimantic uses the node graph to compute probability values which are used to compute the weights assigned to AtomicConcepts in a MixtureConcept. For example, the prior probability of a Concept is proportional to the fraction of edges in the node graph which end on the Concept's corresponding node. Counting these edges can be a time consuming process, so the first time we compute it for a Concept we save the result in the Concept's node. Subsequent operations that rely on this data will be able to quickly load the value instead of counting the edges again. An extra optional compilation step was added that can compute the prior probability of each Concept in the node graph so that the prior probability will always be available at runtime when speed matters.

In the early stages of development, Wikimantic used a node graph constructed from Simple Wikipedia. Simple Wikipedia is a sister site to English Wikipedia, and is written using simple English words to make articles easier for a layperson to understand. The dump files for Simple Wikipedia are almost (but not exactly) the same format as English Wikipedia, and the site as a whole is less popular and therefore much smaller. This makes it an inferior source of information for Wikimantic, but a much better data set for quickly testing basic functionality on weaker machines. Once the basic functionality is there, one can make minor modifications to migrate it to the larger English Wikipedia. This technique was extremely useful for prototyping new ideas and finding bugs, and is recommended to anyone who is thinking of building a system which processes a lot of information from Wikipedia dump files.

8. Conclusion

This paper has presented Wikimantic and its disambiguation and expansion methods that utilize Wikipedia and work with short texts. Our disambiguation method uses a two step process in which topic concepts are hypothesized via a local approach and refined with a global approach. It is robust and performs well on short text fragments in which context words are scarce; it is not limited to nouns, does not rely on correct capitalization, and can determine when a sequence of words should be disambiguated as a single entity. Thus the approach will be useful in retrieval systems that must handle short user queries. Our experimental results show the success of the methodology. Our query expansion methods use the topic concepts produced by Wikimantic for a query and produce very good results for short text documents such as tweets.

There are several aspects of future work. Our experimental results showed that a combination of M and M_R (ReferenceRank) has the potential to improve disambiguation results, but that disproportionate weighting of MixtureConcepts causes the top AtomicConcept to have too much voting power. We plan to explore a smoother method of weighting MixtureConcepts to overcome this problem. We also plan to explore how to selectively use the concept-based query term weighting strategy to improve query expansion and retrieval. Most of all, we plan to use Wikimantic in developing a system for retrieving information graphics in response to user queries, where the query and the text available from the information graphic are short. It will not be possible to evaluate each graphic in detail, so one issue that must be addressed is the fast selection of candidate graphs that will be compared with the user query. We plan to use Wikimantic to expand the terms in the graph and construct an inverted index of graphs using these expanded terms. Then from a user query, we will access the index to quickly select a set of candidate graphs. Another related issue is accounting for entities in the graph that are instances of more general categories in the user query. Consider the query "*Which technology company has the greatest revenue over the last decade*?" and a bar chart whose bars are labeled as *Microsoft*, *Google*, *Yahoo*, etc., but the bar chart does not contain the term *technology company*. Preliminary experiments suggest that Wikimantic will allow us to expand the set of bar labels with the related term *technology company* and thus enable selection of this graph as part of the candidate set that will be further evaluated.

Acknowledgments

This work uses Microsoft Web N-gram Services and was supported by the National Science Foundation under grants III-1016916 and IIS-1017026. We would like to thank the three reviewers for their comments and suggestions which have substantially improved the paper.

References

[1] F. Abel, H. Gao, G. Houben, K. Tao, Semantic enrichment of Twitter Posts for user profile construction on the social web, Proceedings of the 8th Extended Semantic Web Conference on The Semantic Web: Research and Applications, 2011.
[2] B.I Al-Shbou, S. Myaeng, Query phrase expansion using Wikipedia in patent class search, Proceedings of the Asian Information Retrieval Societies Conference, 2011, pp. 115–126.
[3] J. Arguello, J. Elsas, J. Callan, J. Carbonell, Document representation and query expansion models for blog recommendation, Proceedings of the International Conference on Weblogs and Social Media, 2008.
[4] R. Bunescu, M. Pasca, Using encyclopedic knowledge for named entity disambiguation, Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, 2006, pp. 9–16.
[5] R. Burns, S. Carberry, S. Elzer, D. Chester, Automatically recognizing intended messages in grouped bar charts, Proceedings of the 7th International Conference on the Theory and Application of Diagrams, 2012, pp. 8–22.

[6] S. Carberry, S. Elzer, S. Demir, Information graphics: an untapped resource for digital libraries, Proceedings of the 9th International ACM Conference on Research and Development in Information Retrieval, 2006, pp. 581–588.
[7] G. Cao, J. Nie, J. Bai, Integrating word relationships into language models, Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 2005, pp. 298–305.
[8] C. Carpineto, G. Romano, A survey of automatic query expansion in information retrieval, ACM Computing Surveys 20 (1) (2012) 1–50.
[9] J.A. Cohen, A coefficient of agreement for nominal scales, Educational and Psychological Measurement 20 (1) (1960) 37–46.
[10] S. Elzer, S. Carberry, I. Zukerman, The automated understanding of simple bar charts, Artificial Intelligence 175 (2) (2011) 526–555.
[11] A. Fader, S. Soderland, O. Etzioni, Scaling Wikipedia-based named entity disambiguation to arbitrary web text, WikiAI09 Workshop at IJCAI, 2009.
[12] H. Fang, T. Tao, C. Zhai, A formal study of information retrieval heuristics, Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2004.
[13] In: C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, MIT Press, 1998.
[14] P.M. Ferguson, N. OHare, J. Lanagan, A. Smeaton, O. Phelan, K. McCarthy, B. Smyth, Clarity at the Trec 2011 microblog track, Proceedings of the Text Retrieval Conference, 2011.
[15] P. Ferragina, U. Scaiella, TAGME: on-the-fly Annotation of short text fragments (by Wikipedia entities), Proceedings of the 19th ACM International Conference on Information and Knowledge Management, 2010, pp. 1625–1628.
[16] Y. Jing, B. Croft, An association thesaurus for information retrieval, Proceedings of RIAO, 94, 1994, pp. 146–160.
[17] K. Jones, A statistical interpretation of term specificity and its application in retrieval, Journal of Documentation 28 (1) (1972) 11–21.
[18] C. Li, A. Sun, A. Datta, A generalized method for word sense disambiguation based on Wikipedia, Proceedings of the European Conference on Information Retrieval, 2011, pp. 653–664.
[19] Y. Li, R. Luk, E. Ho, F. Chung, Improving weak ad-hoc queries using Wikipedia as external corpus, Proceedings of the International Conference on Research and Development in Information Retrieval, 2007, pp. 797–798.
[20] C. Manning, P. Raghavan, H. Schutze, Introduction to Information Retrieval, Cambridge University Press, 2009.
[21] E. Meij, M. Bron, L. Hollinik, B. Huurnink, M. de Rijke, Learning semantic query suggestions, Proceedings of International Semantic Web Conference, 2009.
[22] E. Meij, M. Bron, L. Hollinik, B. Huurnink, M. de Rijke, Mapping queries to the linking open data cloud: a case study using DBpedia, Web Semantics: Science, Services and Agents on the World Wide Web 9 (4) (2011) 418–433.
[23] E. Meij, W. Weerkamp, M. de Rijke, Adding semantics to microblog posts, Proceedings of Web Search and Data Mining, 2012.
[24] P. Mendes, M. Jakob, A. Garcia-Silva, C. Bizer, DBpedia spotlight: shedding light on the web of documents, Proceedings of the 7th International Conference on Semantic Systems, 2011.
[25] R. Mihalcea, A. Csomai, Wikify!: linking documents to encyclopedic knowledge, Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, 2007, pp. 233–242.
[26] D. Milne, I. Witten, D. Nichols, A knowledge-based search engine powered by Wikipedia, ACM International Conference on Information and Knowledge Management, 2007, pp. 445–454.
[27] D. Milne, I.H. Witten, Learning to link with Wikipedia, Proceedings of the 17th ACM Conference on Information and Knowledge Management, 2008, pp. 509–518.
[28] R. Navigli, Word sense disambiguation: a survey, ACM Computing Surveys 41 (2) (2009) 1–69.
[29] I. Ounis, C. Macdonald, J. Lin, I. Soboroff, Overview of the Trec-2011 microblog track, Proceedings of the Text REtrieval Conference, 2011.
[30] L. Ratinov, D. Roth, D. Downey, M. Anderson, Local and global algorithms for disambiguation to Wikipedia, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011, pp. 1375–1384.
[31] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, Y. Liu, Statistical machine translation for query expansion in answer retrieval, Proceedings of the 45th Meeting of the Association for Computational Linguistics, 2007, pp. 464–471.
[32] I. Ruthven, M. Lalmas, A survey on the use of relevance feedback for information access systems, Knowledge Engineering Review 18 (1) (2003) 95–145.
[33] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford, et al., Okapi at Trec-3, Proceedings of the Text Retrieval Conference, 1995.
[34] H. Schutze, J. Pederson, A co-occurrence based thesaurus and two applications to information retrieval, Information Processing and Management 33 (3) (1997) 307–318.
[35] K. Tao, F. Abel, C. Hauff, WISTUD at TREC 2011: microblog track, Proceedings of the Text Retrieval Conference, 2011.
[36] F. Viegas, M. Wattenburg, D. Kushal, Studying cooperation and conflict between authors with history flow visualizations, Proceedings of International Conference on Computer–Human Interaction, 2004, pp. 575–582.
[37] P. Wu, S. Carberry, S. Elzer, D. Chester, Recognizing the intended message of line graphs, Proceedings of the International Conference on the Theory and Application of Diagrams, 2010, pp. 220–234.
[38] H. Wu, H. Fang, Relation based term weighting regularization, Proceedings of the 34th European Conference on Advances in Information Retrieval, 2012.
[39] J. Xu, B. Croft, Query expansion using local and global document analysis, Proceedings of the International ACM Conference on Research and Development in Information Retrieval, 1996, pp. 4–11.
[40] A. Younos, C. Charles, Frequent itemset mining for query expansion in microblog ad-hoc search, Proceedings of the Text Retrieval Conference, 2012.
[41] W. Zheng, H. Fang, Query aspect based term weighting regularization in information retrieval, Proceedings of the 32nd European Conference on Advances in Information Retrieval, 2010.