Web-Scale Semantic Ranking

Jing Bai
Microsoft Corp
1020 Enterprise Way
Sunnyvale, CA 94089 USA
jbai@microsoft.com

Jan Pedersen
Microsoft Corp
1020 Enterprise Way
Sunnyvale, CA 94089 USA
jpederse@microsoft.com

Mao Yang
Microsoft Research
No.5 Dan Ling Street
Beijing, 100080 China
maoyang@microsoft.com

ABSTRACT

Semantic ranking models go beyond keyword matching to score documents based on closeness in meaning to the query. The use of semantic ranking in Web search has been limited due to the high cost of these models. To address this issue, we have designed and implemented a new Web-scale ranking system that enables us to integrate semantic ranking techniques into a commercial search engine. We have explored several types of models and will describe our implementation of translation models (TM) in this paper. The experiments demonstrate that these models significantly improve relevance over our existing baseline system. Our new ranking system is deployed online and is currently serving many millions of users.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Retrieval Models

General Terms

Algorithms, Performance, Experimentation

Keywords

Web search, Semantic ranking, Forward index, Translation model

1. INTRODUCTION

This paper describes a new Web-scale architecture for the implementation of semantic ranking models. In contrast to traditional keyword matching and ranking systems that depend heavily on an inverted index for performance, our system uses recent innovations in hardware, specifically flash memory drives, to implement a flexible *context-aware* ranking framework on top of a per-document forward index.

Many semantic ranking models have been explored in Information Retrieval (IR) [1][3][4][8][9][12]. These models improve relevance by expending ranking beyond terms in the query. For example, translation models (TM) [3] score text segments based on the probability that the query would have generated the text segment. They have proven to be effective means to improve search quality.

However, traditional implementations of semantic ranking techniques depend heavily on extensive query (or document) expansion [2][7], since the underlying inverted index only supports lookup of individual query terms. This inherent constraint has limited the adoption of semantic ranking models at Web-scale.

In this paper, we describe the design and implementation of a new ranking system that enables the incorporation of full semantic models in a Web-scale search engine. We employ a multi-round ranking architecture where the first matching round uses a traditional inverted index to compute a large number of candidates efficiently using the given query terms, while the second precision-ranking round uses a flexible per-document forward index to consider additional context-dependent expansion terms. Performance in the ranking phase is achieved through the use of high performance flash memory drives to store the forward index and an efficient encoding for the per-document index. This new architecture has numerous advantages including the separation of matching and ranking and the ability to leverage large knowledge models in the ranking phase.

We have explored several types of semantic ranking models: translation models, syntactic pattern matching models and topical matching models. These models are implemented by designing a set of semantic features for input into a machine learned ranking model [6]. Our experiments show that all these models significantly improve relevance over our existing baseline system. We will use translation models as an example in this paper.

In summary, the main contributions of this paper include:

- The design and implementation of a new Web-scale ranking system that enables full semantic ranking models;
- The exploration of several semantic models including translation models and demonstration of significant relevance improvement over our baseline production system.

In the next section, we will describe related work on Web search engine. Then our ranking system design, semantic ranking models and experimental results will be presented. We finish with conclusions and plans for future work.

2. RELATED WORK

Modern search engines use an inverted index to efficiently implement matching and ranking. Ranking models leverage the information stored in inverted index to generate traditional matching and proximity features. A forward index (i.e. a mapping from document to content terms) is only used to generate query-dependent snippets for the top ranking documents.

Massive query expansion, typically through pseudo relevance feedback, is an effective semantic ranking method [1][2]. Several techniques have been proposed to select and re-weight the expansion terms and factor those terms into a document scoring function [1]. However, massive query expansion is difficult to

SIGIR'14, July 6–11, 2014, Gold Coast, Australia.
Copyright 2014 ACM 1-58113-000-0/00/0000 …$15.00.

implement in Web-scale search engines since accessing many expanded terms requires extra runtime cost.

Language models have been successfully applied to IR [10][12]. The basic idea is to rank documents based on the probability the document is relevant given the query. Bayes rule is applied to compute the probability of relevance in terms of the probability of generating the query given the document (i.e. $P(D|Q) \propto P(Q|D)P(D)$) [10]. Different methods are used to compute $P(Q|D)$. Traditional language models use local and global corpus statistics, analogous to *tf* and *idf* [12]. Translation models use fine-grained text alignments to compute these probabilities [3]. For these methods, high quality training data in the form of editorially judged query-document pairs is critical for success. In practice, large-scale training sets of this kind are hard to obtain. Therefore, [3] resorts to the generation of synthetic query-document pairs, and [8] uses implicit relevance judgments from click-through data.

Semantic ranking techniques are not well supported by an inverted index. For example, for a given query, language models may have probability mass on many contextual related terms. If this is implemented through query expansion, pragmatic considerations will likely truncate the model to include only the closest related terms. To address this issue, we have designed and implemented a new Web-scale ranking framework that enables us to integrate full semantic ranking models into a commercial search engine.

3. RANKING SYSTEM DESIGN

Our new ranking system design addresses three main challenges for embedding semantic models into search engines: (1) sufficient memory and computation power for ranking; (2) the efficient access to complete Web documents and knowledge data; and (3) agility in developing new features and models.

3.1 System Overview

A Web search engine finds relevant results for a query from billions of documents. After crawling pages from the Web, an offline document repository stores the documents to be made available online in an inverted and a forward index for future searches. An inverted index maps a term to its occurrences in the indexed corpus, while a forward index maps a document to its contents.

A search engine system consists of a *matching* phase and a *ranking* phase. The matching phase selects and pre-ranks many thousands of documents according to query terms while the ranking phase ranks documents selected by the matching phase using a comprehensive ranking model. To reduce query processing latency, traditional search engines perform matching and ranking on an inverted index that is mostly maintained in memory. In contrast, the forward index, typically stored on disk, is only used to generate query-dependent snippets for a few top ranking documents.

Figure 1 illustrates the overall architecture of our semantic ranking system. Unlike conventional search systems, our new ranking service ② works as an independent service that no longer shares computation resources or the inverted index with the matching service ①. As a result, most of the memory and computation power can be used to enable the implementation of more sophisticated semantic ranking models.

The ranking service works on a per-document index (PDI), which is a type of forward index that can efficiently encode rich semantic information for real-time access during the ranking phase. All information about the document is stored and accessed per document in the PDI whereas an inverted index mostly provides information about the query terms in relation to the document and so ranking models cannot access contextual information. The PDI is implemented as a large-scale distributed table service optimized for fast flash memory storage.

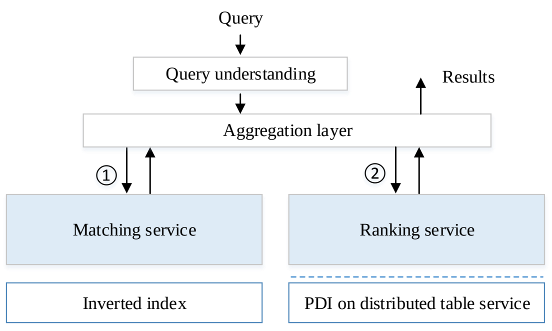

Figure 1. Semantic ranking system architecture

3.2 Per-Document Index

The ranking phase needs to process about 100x more documents from the forward index than conventional search systems which only uses the forward index for snippet generation on top 20 documents. By using flash memory drives which have much higher random access throughput compared to hard disks, the ranking services can process the full contents of thousands of documents.

Figure 2 illustrates our new per-document index design which further addresses this performance challenge through: (1) a well-designed *document model* that allows ranking models to efficiently access different document sections; (2) a brand-new *forward coding* algorithm with improved decoding speed; and (3) an optimized large-scale *table service* to further reduce the cost and latency of index access.

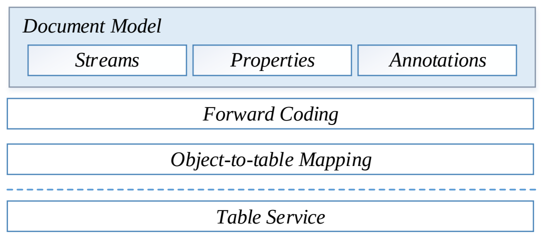

Figure 2. Per-document index

At the core of our ranking platform is the *document model* that provides document information for ranking models. Because not all ranking models need to access all contents of a document at one time, document content is grouped into different sections that can be accessed independently for efficiency. A *stream* section is used to store terms in the document that appear in *Title*, *URL* and *Body* blocks. As stream sections occupy most of the storage, an efficient coding algorithm is used to reduce their size. A *property* section stores document-level metadata, such as the language, location, and creation time of the document. Besides stream and property section types, an *annotation* section is used to enhance basic stream sections with additional information. For example, a *Body* stream only stores a list of basic terms in the body block while other information in the body block, such as punctuation, upper/lower case, etc., is stored in an annotation section.

For the per-document index, we designed a new *forward coding* algorithm that balances compression ratio with decoding speed. The new coding algorithm includes three key features: (1)

dictionary-based pre-processing to reduce the cost of encoding a word; (2) *cache-based* integer coding to further reduce the cost of encoding a word ID; and (3) a *dynamic* approach to reconstruct a word and inverted word list to reduce the chance of decoding all vocabulary for a whole document.

An *object-to-table* module maps document sections to columns of an underlying *table service*. Although the per-document index is logically independent for each document, physically it can be aggregated into one or multiple files on flash memory drives to reduce online serving latency. Sections of a document are grouped into different column groups; e.g. the sections used by Web traffic belongs to one group, while the sections for experimental data belong to another group.

3.3 Semantic Ranking Framework

To reduce the effort of implementing different ranking models for an online system, we design a new ranking framework to facilitate prototyping and development of semantic ranking models. Figure 3 shows our semantic ranking framework. During the ranking phase, each ranking model can access the *query*, the *document*, and the *knowledge store* to generate ranking features for the final learning-to-rank models.

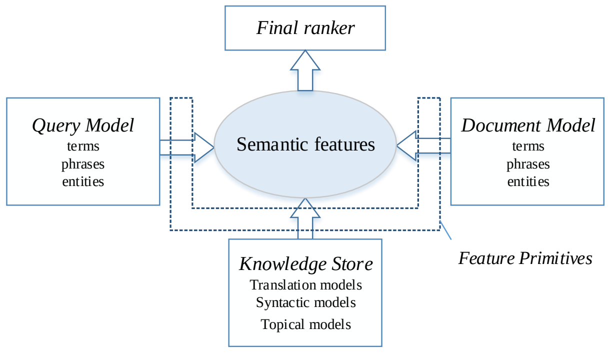

Figure 3. Semantic ranking framework

The *document model* provides a set of operators to access a document, allowing access to every term and to extracted context windows from a stream that contain all or some of the query terms. The *query model* also allows the attachment of semantic information (e.g. entities or phrases) to the original query. Our goal is to construct a better query and document representation in line with the current trend in language modeling [12]. The knowledge store provides data structures for accessing model data in an efficient way, such as a compact in-memory mapping table to store information for translation models.

Rather than developing features from scratch, a set of *feature primitives* allows developers to directly generate features. After semantic features are generated, the ranking system feeds the new features with the traditional keyword-based and other query-independent features into a machine learned ranking model [6] to get the final score for a query-document pair. Section 4 further describes how these new features help improve relevance.

4. SEMANTIC RANKING MODELS

Considering the fact that commercial search engines already incorporate thousands of ranking features and numerous human tweaks, it is challenging to add new features that meaningfully move relevance metrics. Our goal is to design techniques which take full advantage of different types of contextual information at runtime. The main advantages of the PDI stem from the fact that all contextual information about the document is accessed at the time a query is received. During ranking process, the semantic units associated with the query are analyzed and compared to the semantic units within documents in the PDI. Documents that share similar semantics with the query are ranked higher. The use of such information enables the creation of new semantic ranking features which result in relevance improvement. This enables us to move from keyword matching to semantic ranking.

We have explored several types of semantic ranking features but will only describe translation models in this paper. *Statistical machine translation* (SMT) is a machine translation paradigm [3] where translations are generated on the basis of statistical models whose parameters are derived from the analysis of parallel texts (i.e. texts with their translations).

The formula below describes the SMT model. Let $Q = q_1 \dots q_J$ be a query and $D = w_1 \dots w_I$ be the document, the unigram-based translation model [3] assumes that both Q and D are bag-of-words, and the translation probability of a query given a document is calculated as:

$$P(Q|D) = \prod_{q \in Q} \sum_{w \in D} P(q|w) P(w|D) \qquad (1)$$

Here $P(w|D)$ is the unigram probability of word *w* in *D*, and $P(q|w)$ is the probability of translation *w* into a query term *q*.

In our work, the goal is to leverage SMT technologies to improve search relevance, namely by solving the mismatch problem between query and document. The basic idea of translation model is to view queries and documents as being in two different languages, and to bridge the gap between them via translation. For example, if a query contains the term "*software*", a document containing the term "*PowerPoint*" is related. The relationship between terms (i.e. $P(q|w)$) is estimated via a statistical translation model. The model can be trained on different types of parallel texts. In this work, we assume that *queries* are parallel to their frequently clicked document *titles*, and two *click texts* are parallel if both are associated with the same URL (i.e. click-through data from two different user groups).

To learn the translation probabilities, we follow the standard procedure of training statistical word alignment models proposed in [5]. We optimize the model parameters θ by maximizing the probability of generating queries from titles (or generating one click text from others) over the training data. The probability $P(Q|D, \theta)$ takes the form of IBM model 1 in [5].

At ranking time, for a given query, we calculate the translation probabilities from the query to a document using a translation model, and then generate ranking features based on these probabilities. For example, in the *click-based* TM, for a given query-document pair, we go through top click texts of this document and generate new ranking features such as *Max*, *Min*, *Average*, *weighted Average*, *weighted Sum* of their translation probabilities to the query.

The ranking framework is also extended to other TM applications in our work. These include: (1) *bigram* TM, where each source and target contains up to two adjacent terms, (e.g. "ny" => "new york"); (2) *phrase-based* TM, which extracts the key n-grams in the query and document using a learning-to-rank method, and then append them to the initial query and document; and (3) *entity-based* TM, which aims to incorporate more precise concepts that are mined from the internal entity relationship graphs.

5. EXPERIMENTAL RESULTS

In our experiments, we examine the following aspects: (1) What is the relevance improvement resulting from these new semantic ranking features? (2) Is it beneficial to generalize a semantic model to different data sources and to different semantic levels?

In this work, we use *normalized discounted cumulative gain* (NDCG@1) as our primary ranking metric [11]. The experiments train and validate the models using a sampling of editorially judged query-document pairs. In order to test the approach on realistic data, we use data from a commercial search engine in our experiments. We have collected more than 10M editorially judged query-document pairs during a period of two years. We randomly split the data into training set and testing set. *Head* is the test set which contains a natural distribution of queries from our daily traffic, and *Tail* contains a sample of rare queries (e.g. those appear once in the query log).

Each query-document pair is represented by a feature vector. The baseline ranker uses three types of features: (1) *query-based* features; (2) *document-based* features; and (3) *query-document-based* features. Trained on several thousands of features, the baseline ranker was used in our commercial search engine until the semantic ranking system described in this paper went online.

5.1 Applying TM on Different Data Sources

A Web document consists of several fields of information. In order to analyze the contributions from each data source, we conducted several experiments to apply translation model on different streams: *Title*, *URL* and *Click* text.

The results are summarized in Table 1, which are the relative NDCG improvement with respect to our baseline production system due to the application of TM on different text streams. The model we used in this series of tests is a compressed 1GB *unigram* model. A *t*-test is also performed for statistical significance at the level of 95%.

Table 1. Relative NDCG gain using different streams

Stream	Head	Tail	Avg
Title	+0.443	+0.865*	+0.654
URL	+0.377	+0.900*	+0.638
Click	+0.610*	+1.448*	+1.029
Title + URL + Click	+0.812*	+1.683*	+1.247

(* *means significant changes in t-test with respect to the baseline*)

We can see from this table that *Click* produces the best results (+1.029 on average) among all the single streams, and the combination of all data *Title+URL+Click* produces the best results (1.247 on average) among all combinations.

The model used in these experiments is trained on over 1 billion example pairs, which combine data from query-title, query-URL pairs and pairs of queries leading to the same click. Our previous experiments have shown that the universal model (i.e. a model trained using combined data) produces the best results compared to models that are trained separately on different data sources.

5.2 TM to Different Semantic Levels

We generalized the *unigram* TM to other higher-order models as well. Instead of training unigram model on single word, the *bigram* TM is trained from two adjacent terms, the *phrase-based* TM is trained from n-grams (or concepts) extracted by a machine learning approach, and the *entity-based* TM is trained from highlighted entities of the documents. Table 2 shows relative NDCG results of the *bigram,* the *entity-based,* and the *phrase-based* TM respectively.

Table 2. Relative NDCG gain using different models

Model	Head	Tail	Avg
bigram	+0.067	+0.525**	+0.296
phrase-based	+0.233	+0.600**	+0.417
entity-based	+0.500**	+0.800**	+0.650

(** *means significant changes in t-test with respect to the best unigram TM*)

As we can see in Table 2, the *bigram* TM produces some improvements over unigram model, while the *phrase-based* TM produces stronger improvement. The possible reason is that phrases extracted by a reasonable ML method can represent more precise concepts compared to the adjacent terms in bigram model. Among the three models, the *entity-based* TM performs the best, since the precision and coverage of entity model is higher than other models.

6. CONCLUSION

This paper demonstrates for the first time how a full translation model, together with other semantic models, can be employed to significantly improve relevance over a traditional keyword search. To implement semantic ranking techniques, we have designed a new semantic ranking framework to enable the application of full context-aware models. To do this, we implemented a per-document index structure, which explores flash memory drives for latency reduction. The experimental results show significant relevance improvement over our existing system. Many other semantic models can also utilize our semantic ranking framework.

REFERENCES

[1] Bai, J., Nie, J-Y., Bouchard, H., and Cao, G. 2007. Using Query Contexts in Information Retrieval. In SIGIR'07, pp.15-22.

[2] Bai, J., Song, D., Bruze, P., Nie, J-Y., and Cao, G. 2005. Query expansion using term relationships in language models for information retrieval. In CIKM'05, pp. 688-695.

[3] Berger, A. and Lafferty, J. 1999. Information retrieval as statistical translation. In SIGIR'99, pp. 222-229.

[4] Blei, D. M., Ng, A. Y., and Jordan, M. J. 2003. Latent dirichlet allocation. In JMLR, V3, pp. 993-1022.

[5] Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. 1993. The mathematics of statistical machine translation: parameter estimation. In Computational Linguistics, V19 (2), pp. 263-311.

[6] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. 2005. Learning to rank using gradient descent. In ICML'05, pp. 89-96.

[7] Cao, G., Nie, J-Y., and Bai, J. 2005. Integrating word relationships into language models. In SIGIR'05, pp. 298-305.

[8] Gao, J., He, X., and Nie, J-Y. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In CIKM'10, pp. 1139-1148.

[9] Hofmann, T. 1999. Probabilistic latent semantic indexing. In SIGIR'99, pp. 50-57.

[10] Lafferty, J. and Zhai, C. 2001. Document language models, query models, and risk minimization for information retrieval. In SIGIR'01, pp. 111-119.

[11] Jarvelin, K. and Kekalainen, J. 2002. Cumulated gain-based evaluation of IR techniques. In ACM Transactions on Information Systems, V20, pp. 422-446.

[12] Ponte, J. and Croft, W. B. 1998. A language modeling approach to information retrieval. In SIGIR'98, pp. 275-281.