Foundations and Trends® in Information Retrieval
Vol. 7, No. 5 (2013) 343–469

DOI: 10.1561/1500000035

now
the essence of knowledge

Semantic Matching in Search

Hang Li
Huawei Technologies, Hong Kong
hangli.hl@huawei.com

Jun Xu
Huawei Technologies, Hong Kong
nkxujun@gmail.com

Contents

Abstract

Relevance is the most important factor to assure users' satisfaction in search and the success of a search engine heavily depends on its performance on relevance. It has been observed that most of the dissatisfaction cases in relevance are due to term mismatch between queries and documents (e.g., query "ny times" does not match well with a document only containing "New York Times"), because term matching, i.e., the bag-of-words approach, still functions as the main mechanism of modern search engines. It is not exaggerated to say, therefore, that mismatch between query and document poses the most critical challenge in search. Ideally, one would like to see query and document match with each other, if they are topically relevant. Recently, researchers have expended significant effort to address the problem. The major approach is to conduct semantic matching, i.e., to perform more query and document understanding to represent the meanings of them, and perform better matching between the enriched query and document representations. With the availability of large amounts of log data and advanced machine learning techniques, this becomes more feasible and significant progress has been made recently. This survey gives a systematic and detailed introduction to newly developed machine learning technologies for query document matching (semantic matching) in search, particularly web search. It focuses on the fundamental problems, as well as the state-of-the-art solutions of query document matching on form aspect, phrase aspect, word sense aspect, topic aspect, and structure aspect. The ideas and solutions explained may motivate industrial practitioners to turn the research results into products. The methods introduced and the discussions made may also stimulate academic researchers to find new research directions and approaches. Matching between query and document is not limited to search and similar problems can be found in question answering, online advertising, cross-language information retrieval, machine translation, recommender systems, link prediction, image annotation, drug design, and other applications, as the general task of matching between objects from two different spaces. The technologies

introduced can be generalized into more general machine learning techniques, which is referred to as learning to match in this survey.

H. Li and J. Xu. *Semantic Matching in Search*. Foundations and Trends® in Information Retrieval, vol. 7, no. 5, pp. 343–469, 2013.

DOI: 10.1561/1500000035.

1

Introduction

1.1 Query Document Mismatch

A successful search engine must be good at relevance, coverage, freshness, response time, and user interface. Among them, relevance [156, 171, 157] is the most important factor, which is also the focus of this survey.

This survey mainly takes general web search as example. The issues discussed are not limited to web search, however; they exist in all the other searches such as enterprise search, desktop search, as well as question answering.

Search still heavily relies on the bag-of-words approach or term based approach. That is, queries and documents are represented as bags of words (terms), documents are indexed based on document terms, 'relevant' documents are retrieved based on query terms, the relevance scores between queries and retrieved documents are calculated on the basis of matching degrees between query terms and document terms, and finally the retrieved documents are ranked based on the relevance scores. This simple approach works quite well in practice and it still forms the foundation of modern search systems [131, 52, 6].

Table 1.1: Examples of query document mismatch.

query	document	term match	semantic match
seattle best hotel	seattle best hotels	partial	yes
pool schedule	swimming pool schedule	partial	yes
natural logarithm transform	logarithm transform	partial	yes
china kong	china hong kong	partial	no
why are windows so expensive	why are macs so expensive	partial	no

The bag-of-words approach also has limitations, however. It sometimes suffers from the query document mismatch drawback. For the majority of the cases of dissatisfaction reported at a commercial web search engine, in which users complain they cannot find information while the information does exist in the system, the reasons are due to mismatch between queries and documents. Similar trends are observed in other studies (cf., [206, 207])

A high matching degree at term level does not necessarily mean high relevance, and vice versa. For example, if the query is "ny times" and the document only contains "New York Times", then the matching degree of the query and the document at term level is low, although they are relevant. More examples of query document mismatch are given in Table 1.1.[1]

Query document mismatch occurs, when the searcher and author use different terms (representations) to describe the same concept, and this phenomenon is prevalent due to the nature of human language, i.e., the same meaning can be represented by different expressions and the same expression can represent different meanings. According to Furnas et al., on average 80-90% of the times, two people will name the same concept with different representations [67].

[1] China Kong is an American actor.

Table 1.2: Queries about "distance between sun and earth".

"how far" earth sun	average distance from the earth to the sun
"how far" sun	how far away is the sun from earth
average distance earth sun	average distance from earth to sun
how far from earth to sun	distance from earth to the sun
distance from sun to earth	distance between earth and the sun
distance between earth & sun	distance between earth and sun
how far earth is from the sun	distance from the earth to the sun
distance between earth sun	distance from the sun to the earth
distance of earth from sun	distance from the sun to earth
"how far" sun earth	how far away is the sun from the earth
how far earth from sun	distance between sun and earth
how far from earth is the sun	how far from the earth to the sun
distance from sun to the earth	

Table 1.2 shows example queries representing the same search need "distance between sun and earth" and Table 1.3 shows example queries representing the same search need "Youtube", collected from the search log of a commercial search engine [117]. Ideally, we would like to see the search system return the same results for the different variants of the queries. Web search engines, however, still cannot effectively satisfy the requirement. This is another side of the mismatch problem.

In web search, query document mismatch more easily occurs on tail pages and tail queries. This is because for head pages and head queries, usually there is more information attached to them. A head page may have a large number of anchor texts and associated queries in search log and they provide with the page different representations. The matching degree will be high, if the query matches with any of the representations. This seldom happens to a tail page, however. Mismatch, thus, is a typical example of the long tail challenge in search.

1.2 Semantic Matching in Search

The fundamental reason for mismatch is that no language analysis is conducted in search. Language understanding by computer is hard,

Table 1.3: Queries about "Youtube".

yutube	yuotube	yuo tube
ytube	youtubr	yu tube
youtubo	youtuber	youtubecom
youtube om	youtube music videos	youtube videos
youtube	youtube com	youtube co
youtub com	you tube music videos	yout tube
youtub	you tube com yourtube	your tube
you tube	you tub	you tube video clips
you tube videos	www you tube com	wwww youtube com
www youtube	www youtube com	www youtube co
yotube	www you tube	www utube com
ww youtube com	www utube	www u tube
utube videos	utube com	utube
u tube com	utub	u tube videos
u tube	my tube	toutube
outube	our tube	toutube

however, if not impossible. A more realistic approach beyond bag-of-words, referred to as semantic matching in this survey, would be to conduct more query analysis and document analysis to represent the meanings of the query and the document with richer representations and then perform query document matching with the representations. The analysis may include term normalization, phrase analysis, word sense analysis, topic analysis, and structure analysis, and the matching may be performed on form aspect, phrase aspect, word sense aspect, topic aspect, and structure aspect, as shown in Figure 1.1. Intuitively, if the meanings of the query and the document represented by the aspects are the same, then they should match each other well and thus be regarded relevant. In practice, the more aspects of the query and document can match, the more likely the query and document are relevant. With semantic matching, we can expect that the query document mismatch challenge can be successfully conquered.

Term normalization, including word segmentation for Asian languages, compounding for European languages, spelling error correction for European languages, should usually be carried out before query document matching. We refer to term normalization as matching on the form aspect. Query document matching on the phrase aspect means that the two should match at phrase level, not word level. For example, if the query is "hot dog", then it should be recognized as a phrase and match the exactly same phrase in the document, but should not separately match words "hot" and "dog" in the document. Matching on the word sense aspect is to have phrases in the query and the document having the same sense match each other. For example, "ny" should match "New York". If the query and the document have the same topics, then they should match on the topic aspect. For example, if the query is "microsoft office" and the document is about Microsoft Word, PowerPoint, and Excel, then the two should match in terms of topic. Query and document can also match on the structure aspect, where structure means linguistic structure. For example, the query "distance between sun and earth" matches with the document title "how far is sun from earth" (note that the two expressions have very different linguistic structures).

We can also consider query document matching on other aspects, for example, semantic class and named entity. We will discuss this in Section 9 on conclusion and open problems.

Semantic matching is also a term used in other fields in computer science, which represents a notion different from this survey. Given two graph-like structures, e.g., two database schemas, semantic matching is defined as an operator that identifies the nodes in the two structures which semantically correspond to each other [73].

Semantic matching also differs from the so-called semantic search, which has different definitions by different researchers. One of them is aimed at enriching search results of a conventional search system, by using information from semantic web (e.g., [77]). For example, the search result of query "yo-yo ma" is augmented by the cellist's image, concert schedule, music albums, etc. in the semantic search. The semantic search by Bast et al. asks the user to formulate a

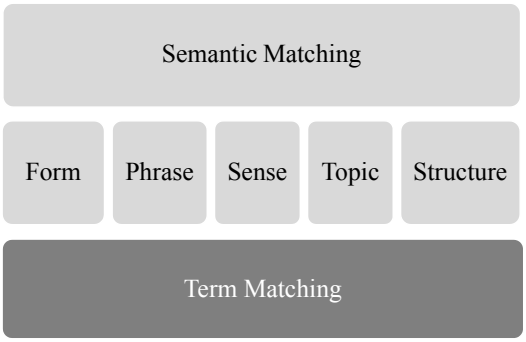

Figure 1.1: Semantic matching: if the meanings of the query and document represented in the aspects of form, phrase, sense, topic, and structure are the same, then they should match each other and be regarded relevant.

query with operators describing relations between entities, combines the information found from both documents and ontology, and returns to the user. Special search needs such as "finding plants with edible leaves and native to Europe" are supported [11]. In contrast, the semantic matching which we are concerned with here is carried out inside the search engine and users do not need to do anything different from conventional search.

Figure 1.2 illustrates the difference between semantic matching and semantic search. Semantic matching is concerned with search of documents by query, where both documents and query are unstructured data. Semantic search is usually concerned with search of documents and knowledge base by query, where documents and query are unstructured data, but knowledge base is structured data.

Query document mismatch has been studied in the long history of information retrieval (IR). In traditional IR, methods such as query expansion, pseudo-relevance feedback, and latent semantic indexing (LSI) have been intensively investigated and widely utilized. Nowadays large amounts of log data have been collected in web search and advanced machine learning techniques have been developed. We can really leverage big data and machine learning to more effectively

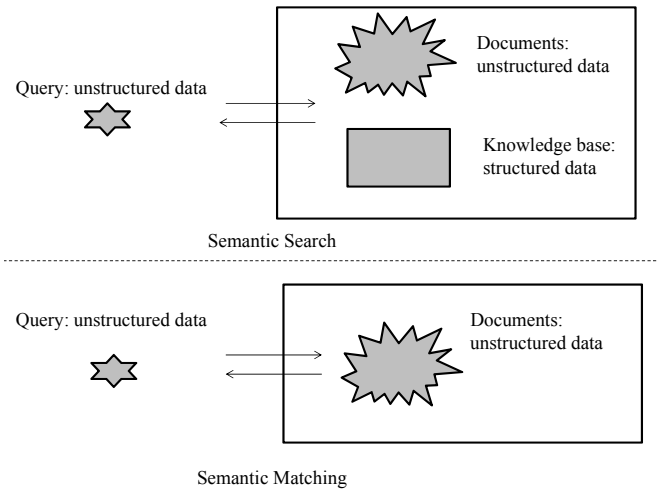

Figure 1.2: Semantic matching versus semantic search.

address the challenge of query document mismatch, as explained in this survey.

1.3 Matching and Ranking

In traditional IR, the distinction between ranking and matching in search is not made clear. Given a query, documents are retrieved from the index and matching between the query and each of the documents is carried out. The relevance of the document with respect to the query is represented as the matching degree between the two, calculated using an IR model (matching model) such as BM25 or language models for information retrieval (LM4IR). After that, the documents are ranked (sorted) based on their matching scores. In such a framework, matching scores and ranking scores are equivalent. [2]

Things have changed in web search. Importance of documents (web pages) is found useful for relevance ranking, and importance scores of

[2]We note that in modern web search not only relevance but also freshness, diversity, and other factors are considered. We restrict ourselves to relevance in this survey.

web pages calculated by models such as PageRank need to be incorporated into the ranking mechanism. Besides, many signals indicating the relevance (matching) degrees between queries and documents are also available and matching scores representing the signals can be calculated. How to combine the matching scores and importance scores then becomes a critical question. A simple approach is to linearly combine the scores and manually tune the weights. More sophisticated machine learning techniques for automatically constructing the ranking model using training data can also be considered. In fact, machine learning techniques for the purpose, referred to as learning to rank, have been intensively studied and widely applied in web search [128, 115]. Thus, in web search, the processes of matching and ranking are logically separated (first matching and then ranking).

As explained below, machine learning techniques for learning matching degrees between queries and documents (in general, heterogeneous objects), which are referred to as learning to match in this paper, have been developed. Learning to match is in fact *feature learning* for learning to rank, from the viewpoint of machine learning.

1.4 Semantic Matching in Other Tasks

Other tasks in information retrieval and natural language processing also rely on *semantic matching*, such as paraphrasing & textual entailment [62, 54], question answering [21], cross-language information retrieval (CLIR) [141, 140], online advertising [31], similar document detection [32, 33], and short text conversation [176, 130]. Table 1.4 summarizes the characteristics of the tasks.

For instance, CLIR is a subfield of information retrieval concerning with the problem of receiving queries in one language while retrieving documents in another language. Translation of either query or document from one language to another is naturally required in the task. Mismatch between query and document in two languages poses an even greater challenge to CLIR and matching on form aspect (compounding, word segmentation, spelling error correction), sense aspect (selection

of translation), and topic aspect has also been tried and verified to be helpful [141, 140].

For another instance, online advertising makes use of web to deliver marketing messages and attract consumers. It usually involves publishers, who display advertisements at their web sites, and advertisers, who provide advertisements. Given some advertisements, it is necessary to find appropriate web sites for displaying them, i.e. conduct effective matching between publishers' content and advertisers' advertisements. Mismatch is also inevitable here. Methods have been proposed for addressing the mismatch challenge at form aspect, sense aspect, and topic aspect [31].

Short text conversation is a research problem proposed recently [176, 130]. It consists of one round of conversation between human and computer, with the former being a message from human and the latter being a comment on the message from the computer. Short text conversation constitutes one step of natural language conversation, and it also subsumes question answering as special case. Semantic matching between messages and comments needs also be considered, in a retrieval based approach in which a large collection of message and comment pairs is indexed, and given a message the most appropriate comment is retrieved, selected, and returned. Methods have been proposed to address the mismatch problem in the task as well [176, 130].

1.5 Machine Learning for Semantic Matching in Search

A natural question arises whether it is possible to use machine learning techniques to automatically learn the models for semantic matching in search. This is exactly the problem we address in this survey.

The task can be formalized as learning of matching model $f(q, d)$ or conditional probability model $P(r|q, d)$ using supervised learning techniques or learning of conditional probability model $P(q|d)$ using unsupervised learning techniques, where q denotes query, d denotes document, and r denotes relevance level. Note that here query and document are regarded as different (heterogeneous) objects.

Table 1.4: Characteristics of tasks that need semantic matching. Two natural language texts (A and B) are involved in the tasks.

task	types of texts	relation between texts
search	A=query, B=document	relevance
question answering	A=question, B=answer	answer to question
cross-language IR	A=query, B=document (in diff. lang.)	relevance
short text conversation	A=text, B=text	message and comment
similar document detection	A=text, B=text	similarity
online advertising	A=query, B=ads.	relevance as ads.
paraphrasing	A=sentence, B=sentence	equivalence
textual entailment	A=sentence, B=sentence	entailment

Different models can be defined, explicitly or implicitly representing semantic matching, i.e., matching on different aspects such as form aspect, phrase aspect, sense aspect, topic aspect, and structure aspect. Since query document mismatch is a long tail phenomenon, it is necessary to assume that no single signal is enough and construct matching models on different aspects and combine the uses of them in relevance ranking.

The following are some well-studied approaches, including matching by query reformulation, matching with term dependency model, matching with translation model, matching with topic model, and matching with latent space model. This survey will explain the approaches in detail.

Matching by query reformulation aims at reformulating the query so that it can have a better match with the semantically equivalent expressions in the documents. Reformulation of query includes spelling

error correction, word splitting, word merging, and so on. The major issues with regard to query reformulation include re-writing of the original query, blending of the search results by the original query and reformulated queries, mining of similar queries, as well as query expansion.

A straightforward extension of the bag-of-words approach would be to perform matching based on multiple words in the query and document. This is exactly the process depicted in the term dependency models. One can represent different matching relations between the query terms and the document terms with the models, for example, co-occurrence of terms in both the query and document. Intuitively, if several terms co-occur within both the query and document, then they may represent the same concept and indicate stronger relevance.

Matching between the query and a part of the document, for example, the title, can be modeled as paraphrasing or translation in which a language expression is transformed into another language expression. Taking matching as a statistical translation task has been proposed previously and the approach has made significant progress in web search recently, in part because a large amount of click-through data becomes available and can be utilized as training data.

Given a collection of documents, topic modeling techniques can help find the topics of the documents, in which each topic is represented by a number of words. Probabilistic and non-probabilistic models have been proposed. In search, the topics of the query and the topics of the documents can be detected, and matching between the query and documents can be carried out with the topics.

We can represent queries and documents in two different vector spaces, map them into a hidden semantic space with lower dimensionality on the basis of query document associations in click-through data, and conduct matching between queries and documents in the latent semantic space. This is the basic idea of the approach of matching with latent space models. Many traditional IR models such as vector space model (VSM), BM25, and LM4IR can be interpreted as special cases of the latent space models, and thus the latent space models are quite fundamental for IR.

Matching between two heterogenous objects is not limited to search. It exists in many other applications, including paraphrasing & textual entailment, question answering, online advertising, cross-language information retrieval, similar document detection, short text conversation, machine translation, recommender systems (collaborative filtering), link prediction, image annotation, and drug design. It is necessary and important to generalize the techniques developed in different applications to a more general machine learning methodology in order to study the techniques more deeply and broadly. We refer to it as learning to match in this survey.

1.6 About This Survey

This survey focuses on the fundamental problems, as well as the state-of-the-art solutions of query document matching in search. The ideas and solutions explained may motivate industrial practitioners to turn the research results into products. The methods introduced and the discussions made may also stimulate academic researchers to find new research directions and approaches.

Section 2 gives a formulation of machine learning for query document matching in search and shows an implementation of it in web search. Sections 3-7 describe the five groups of learning techniques for query document matching, namely matching by query reformulation, matching with term dependency model, matching with translation model, matching with topic model, and matching with latent space model. Section 8 describes generalization of the techniques, learning to match, and introduce methods for collaborative filtering and paraphrasing & textual entailment. Section 9 summarizes the survey and discusses open problems. Sections 2-8 are self-contained, and thus the reader can choose sections to read on the basis of their interest and need.

This survey focuses more on machine learning and semantic matching. Several survey papers or books cover some related topics, such as LM4IR [204], query expansion [40], search and browse log

mining [163, 94], and feature centric view on IR [135]. The reader is also encouraged to refer to the materials.

We assume that the reader has certain knowledge on machine learning and information retrieval. Those who want to know more about the fundamentals of the areas should refer to related books and papers. The machine learning techniques concerned with in this survey include statistical language model [204], statistical machine translation [99], learning to rank [128, 115, 116], graphical model [24], topic model [25], matrix factorization [103], kernel methods [158], sparse methods [3], and deep learning [4]. Explanations on the basic techniques in information retrieval can be found in the text books on IR [131, 52, 6].

[3] A tutorial on sparse methods by Bach can be found at www.di.ens.fr/fbach/.
[4] Tutorials on deep learning can be found at www.deeplearning.net/tutorial/.

2

Semantic Matching in Search

This section gives a mathematical view on machine learning for query document matching, i.e., semantic matching. It also gives a system view on semantic matching.

2.1 Mathematical View

Learning for query document matching can be performed in both supervised learning setting and unsupervised learning setting. Here, supervised learning means that the responses (matching degrees) of query document pairs are given in the learning phase, while unsupervised setting means that they are not.

2.1.1 Supervised Learning

Suppose that training data (q_1, d_1, r_1), (q_2, d_2, r_2), $\cdots$, (q_N, d_N, r_N) is given. Each sample is a triple representing query q, document d, and response r. The response represents the matching degree (relevance) between the query and the document. Query q is generated according to probability distribution $P(q)$, document d is generated according to conditional probability distribution $P(d|q)$, and response r

is generated according to conditional probability distribution $P(r|q,d)$. This corresponds to the following fact. Queries are submitted to the search system independently, documents are retrieved with the query words given a query, and the relevance can be approximately determined given a pair of query and document. Click through data collected at a search engine can be used as the training data.[1]

The goal of learning to match is to automatically learn a model represented either as function $f(q,d)$ or as conditional probability distribution $P(r|q,d)$. (Note that $f(q,d)$ is a more general definition, because one can define $f(q,d) = P(r|q,d)$. Also note that the use of $P(r|q,d)$ in relevance ranking has been studied in IR from many years ago [152].)

The matching problem is similar to conventional classification and regression problems. There is also dissimilarity, however. Classification and regression are about learning of a one-input function (i.e., a function of one feature vector) and matching is about learning of a two-input function (i.e., a function of two feature vectors). The relations between the inputs can be and should be leveraged in the matching task.

The learning problem can be formalized as minimization of the following regularized empirical loss function

$$\min_f \sum_{i=1}^{N} L(r_i, f(q_i, d_i)) + \Omega(f),$$

where $L(r, f(q,d))$ denotes a loss and $\Omega(f)$ denotes a regularization.

As explained, semantic matching between queries and documents can be conducted on word sense, topic, and structure aspects. That means different matching functions can be learned within the above learning framework.

Here are two example formulations. (1) ID matching: q and d are just represented as IDs. Matching between queries and documents can be formalized as matrix factorization, under a strong assumption that the queries and documents are fixed. (2) Feature matching: q and

[1] Click-through data is noisy. One common method for data cleaning is threshold cut-off [94]. More research on the problem is surely necessary.

d are represented as feature vectors. Matching between queries and documents can be formalized, for instance, as feature-based matrix factorization. In fact, the latent space models described in Section 7 are specific instances of (2).

The challenges of the learning task include the large scale of problem and the sparseness of training data.

2.1.2 Unsupervised Learning

The matching degree between query q and document d can be represented by the conditional probability $P(q|d)$.

One view on matching and ranking with this approach is as follows. Given a query, we retrieve documents and calculate the conditional probabilities (matching scores) of documents with respect to the query, represented as $P(d|q)$, and then rank the documents according to the conditional probabilities $P(d|q)$. By Bayes' rule we have

$$P(d|q) \propto P(d)P(q|d),$$

where $P(d)$ denotes the prior probability (importance score) of document d.[2] $P(d)$ can be calculated, for example, using PageRank [143]. The question then is how to calculate $P(q|d)$.

In the language model approach, query q and document d are usually viewed as bags of words (unigrams). We refer the reader to [204] for detailed explanation of LM4IR. Suppose that documents $d_1, d_2, \cdots, d_N$ are given. Each document d_i is represented as $w_1^i, w_2^i, \cdots, w_{|d|}^i$. The conditional probability $P(q|d_i)$ is calculated based on the data of q and the distribution of $P(w|d_i)$, where w denotes a word in the vocabulary. That is the likelihood of query q's being generated from document d_i.

The estimation of $P(w|d_i)$ can be performed by maximum likelihood estimation (MLE). (For simplicity, we do not consider smoothing of the probabilities. See [204] for various smoothing techniques.) We have

$$P(w|d_i) = \frac{f(w, d_i)}{|d_i|},$$

[2]The probability $P(d|q)$ can also be calculated by a learning to rank model, where the probabilities $P(q|d)$ and $P(d)$ are utilized as features of the model [115].

where $f(w, d_i)$ denotes the frequency of word w occurring in document d_i and $|d_i|$ denotes the length of document d_i. Hence, the conditional probability $P(q|d_i)$ can be calculated as follows.

$$P(q|d_i) = \prod_{k=1}^{|q|} P(w_k^q|d_i),$$

where $P(w_k^q|d_i)$ denotes the probability that query word w_k^q is generated from document d_i.

Although it is effective, the bag-of-words approach suffers from the mismatch problem. Several approaches can be considered to deal with the challenge. In the topic modeling approach, for example, it is assumed that the words are generated from a topic model

$$P(w|d_i) = \sum_z P(z|d_i)P(w|z),$$

as described in Section 6, where z denotes a topic, $P(z|d_i)$ the probability of topic z given document d_i, and $P(w|z)$ the probability of word w being generated from topic z. Through topics, the matching between query and document can be based on meaning rather than word.

The challenges for this learning task also lie in the large scale of problem and the sparseness of training data.

2.2 System View

We next describe how to perform semantic matching in web search. Figure 2.1 shows a web search system that conducts semantic matching. It includes crawling, document understanding, indexing, query understanding, retrieving, query document matching, and ranking modules, as well as index and user interface. Figures 2.2, 2.3, and 2.4 further show the processing in the query understanding, document understanding, and (semantic) matching modules. See [131, 52, 6] for introduction to the fundamentals of web search system.

Query understanding consists of spelling error correction, phrase identification, similar query finding, topic identification, and structure

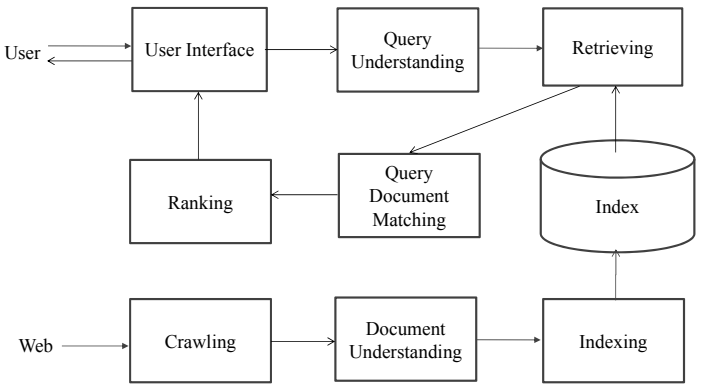

Figure 2.1: Architecture of web search engine.

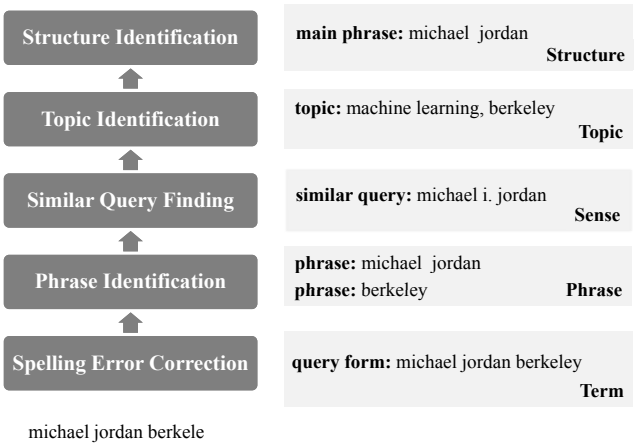

Figure 2.2: Query understanding.

identification. Document understanding consists of phrase identification, key phrase identification, topic identification, and structure identification. Matching between richer query and document representations is then performed, which realizes semantic matching between query and document. The matching employs the models learned in advance, on the aspects of form, phrase, sense, topic, and structure.

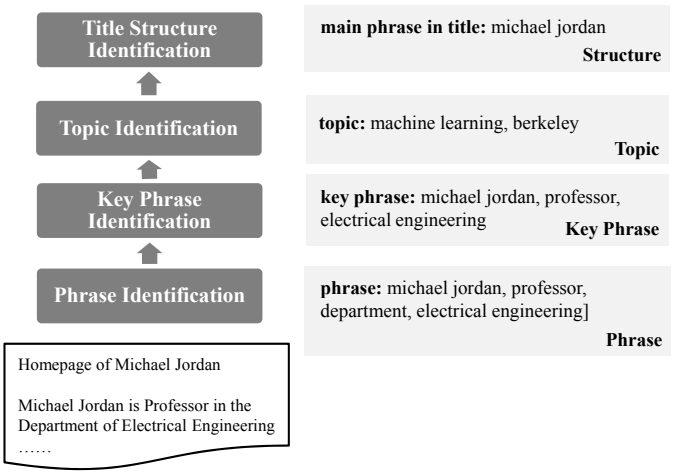

Figure 2.3: Document understanding.

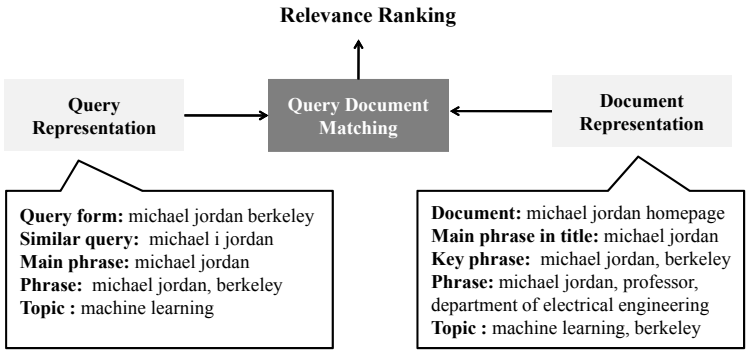

Figure 2.4: Query document matching.

Figure 2.2 shows an example of query understanding. Suppose that the input is "michael jordan berkele", which contains a typo. Spelling error correction is first made on the query. Phrases in the query are then identified as "michael jordan" and "berkeley". Next, similar queries such as "michael i jordan" are retrieved. Topics of the query such as "machine learning" are also identified. Finally, the structure of the query is analyzed with "michael jordan" identified as key phrase. Note

that the first step is typically performed first, but the other steps can be performed independently. We put them into a pipeline only for ease of explanation.

Figure 2.3 shows an example of document understanding. Suppose that the homepage of Michael Jordan at Berkeley is given. First, phrases such as "Michael Jordan" and "professor" are identified. Next, the key phrases are selected. Topics of the page such as "Berkeley", "machine learning" are also identified. Finally, the structure of the page, including the structure of the title is recognized. Again, the steps can be executed independently.

Figure 2.4 shows an example of query document matching. The enriched query and document representations are matched to determine the relevance of the document (webpage) with respect to the query.

3

Matching by Query Reformulation

When mismatch between query and document occurs, a straightforward way to tackle the problem would be to transform the query to another query which can better represent the search need and can make better match with relevant documents. This is exactly the proposition behind the approach of matching by query reformulation, which in some sense has the longest history as a method of dealing with mismatch in IR. Query reformulation includes spelling error correction, stemming, query segmentation, query expansion, and query deduction. (For Asian languages, it is usually assumed that word segmentation has been conducted on query and document before the process.) This section first gives an overview of query reformulation, and then describes four issues with regard to query reformulation, including methods of query reformulation, similar query mining, search result blending, and query expansion. It also shows some experimental results. Similar query mining is about automatic discovery of similar queries from search log and web data, which are then utilized in query reformulation. Search result blending is concerned with fusion of search results retrieved by both the original query and reformulated queries. Query expansion is

Table 3.1: Types of query reformulation.

type	example
spelling error correction	mlss singapore → miss singapore
merging	face book → facebook
splitting	dataset → data set
stemming	seattle best hotel → seattle best hotels
synonym	ny times → new york times
segmentation	new work times square → "new york" "times square"
query expansion	www → www conference
query deduction	natural logarithm transformation → logarithm transformation
stopword removal\preservation	the new year → "the new year" [1]
paraphrasing	how far is sun from earth → distance between sun and earth

related to adding new terms to the original query (expanding the query) and retrieving with the expanded query.

3.1 Query Reformulation

Query reformulation, also referred to as query re-writing, query transformation, query refinement, and query alteration, is to transform the original query to queries (representations) that can better match with documents, in the sense of relevance [51]. Suppose that the query is "ny times" and the document contains "New York Times". If we reformulate the query into "new york times", then we will be able to match the (reformulated) query to the document and solve the mismatch problem. Table 3.1 gives the major types of query reformulation.

The main challenge to query reformation is topic drift. It is hard to ensure that a reformulated query represents the same search need as the

[1] "The new year" is the title of an American movie, and thus the word "the" should not be removed here, although it is usually treated as stopword.

original query. In the worst case, reformulation will result in creating a query with a very different meaning. For example, with stemming, "arms reduction" could be transformed to "arm reduction" and the use of the reformulated query could hurt relevance.

Another challenge is that there is no guarantee that search relevance can be improved by query reformulation. It is a task also depending on the content of the document collection. For example, if the query is "seattle best hotel" and the documents do not contain "Seattle best hotels", then transforming the former to the latter would not help enhance relevance. At the query understanding phase, it is usually impossible to judge whether a particular query reformulation would help. It is only after matching the fact becomes evident.

Nonetheless, it has been observed that spelling error correction, definite splitting (e.g., "united states"), and definite merging (e.g., "facebook") can help improve relevance [79]. Furthermore, query expansion by pseudo-relevance feedback can achieve state-of-the-art performances on conventional IR datasets [40].

3.2 Methods of Query Reformulation

3.2.1 Spelling Error Correction

Usually about 10-15% of English queries contain spelling errors, which becomes one of the major issues preventing users from finding information (e.g., [53]). It is critically important, therefore, to transform misspelled queries into well-formed queries and help users to quickly find the information they look for.

For simplicity let us first consider correction of individual misspelled words (e.g., "elefnat" to "elephant"). One simple approach to spelling error correction is to calculate the edit distance between the query word and each of the dictionary words. Dictionary words within a fixed range of edit distance or a variable range of edit distance depending on word length are selected as candidates for correction. There are at least two drawbacks for this approach, however. First, probabilities of word usages as well as word misspellings are not considered in the model. For example, people rarely mistype "antler" as "entler", but often mistype

"reluctant" as "reluctent". Second, context information of correction is not taken into consideration. For example, people tend to misspell "c" to "s" or "k" depending on contexts. The use of edit distance cannot deal with the problems.

To address the issues, probabilistic approaches, both generative approach and discriminative approach, have been proposed. Suppose that the query word is represented as q and a correction is represented as c. We want to find the correction $\hat{c}$ having the largest conditional probability $P(c|q)$. Different ways of defining the model lead to different methods.

By Bayes' rule, we can consider finding the correction $\hat{c}$ having the largest product of probability $P(c)$ and conditional probability $P(q|c)$

$$\hat{c} = \arg\max_c P(c|q) = \arg\max_c P(c)P(q|c).$$

This is the basic idea behind the generative approach. The former is called source model and the latter channel model. An interpretation of using the models is that the intended query word has been distorted through transmission of it via a noisy channel and we want to recover the intended query word from the observed distorted (misspelled) query word. One advantage of the generative approach is that we can separately train the source and channel models. Different generative methods in fact employ different formulations of the source model and channel model.

The source model can be trained by using the document collection and/or search log. (Due to the wide variety of searches it is better to find the *legitimate* words from data.) A straightforward way would be to estimate the probabilities of words based on their occurrences in the dataset with a smoothing technique applied.

The channel model can be defined based on weighted edit distance, where the model is usually trained by using data consisting of pairs of correct word and misspelled word (cf., [151, 3]). We next describe two methods for constructing the channel model.

Brill and Moore [29] have developed a method for building a source channel model for spelling error correction. The major characteristics of their method is to use contextual substitution (transformation) rules, in other words, to perform substitutions of substrings on the

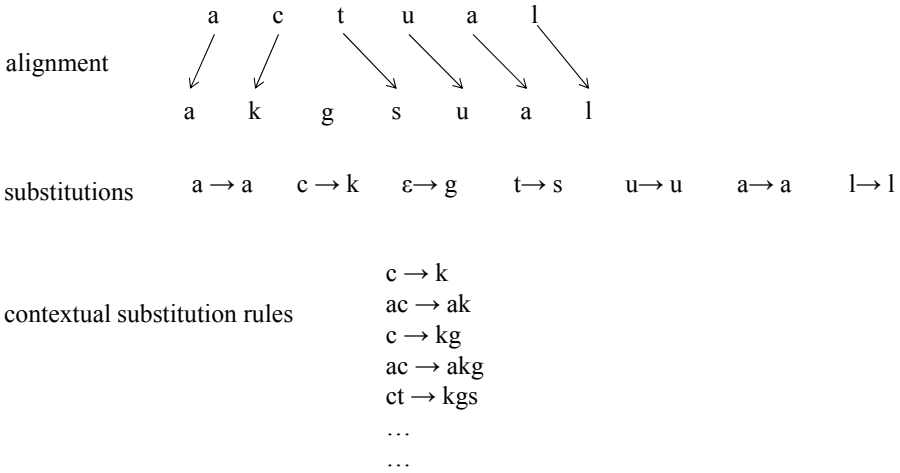

Figure 3.1: Examples of contextual substitution rules.

basis of contexts. Given pairs of misspelled word and correct word as training data, their method first finds the 'best' alignment of segments between the two words in each pair on the basis of edit distance, derives substitution rules from the alignments, and heuristically estimates the probabilities of the substitution rules. Figure 3.1 shows an example of substitution rule derivation. In spelling error correction, they utilize a trie to store the dictionary and another trie to store the rule set to efficiently find the corrections with the largest total probabilities. Toutanova and Moore [172] have further improved the model by adding pronunciation factors into it, and Cucerzan and Brill [53] have developed a method for iteratively training the model using query log.

Duan and Hsu's method [63] takes the query word and the correction as a sequence of 'transfemes', and defines the channel model based on the sequence, where a transfeme denotes a transformation of one substring to another substring between the query word and the correction. Figure 3.2 shows an example of sequence of transfemes. Given a specific segmentation of query and correction, an n-gram model on the sequence of transfemes can be defined. The channel model is

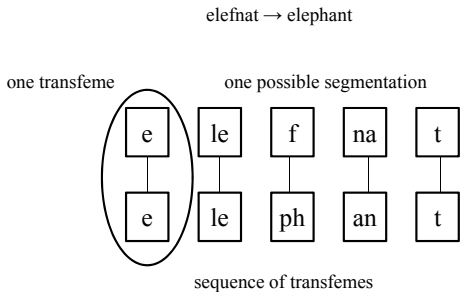

Figure 3.2: Example of sequence of transfemes.

defined as a probability model over all the possible segmentations.

$$P(q|c) = \sum_{s \in S(c \to q)} \prod_{i}^{l(s)} P(t_i | t_{i-n+1}, \cdots, t_{i-1}),$$

where c denotes a correct word, q a misspelled word, $S(c \to q)$ the set of segmentations for c and q, s a segmentation, $l(s)$ the length of s, t a transfeme, and n the number of transfemes. The segmentations are not observable in the training data, however. Their method employs the expectation maximization (EM) algorithm to estimate the probabilities of the channel model. In online prediction, the method utilizes a trie for indexing of the dictionary and uses the A* algorithm for efficient retrieval of correction candidates.

Wang et al. [183] propose a discriminative method for spelling error correction, which can be viewed as a counterpart of Brill and Moor's generative method. Their method utilizes the conditional probability model $P(c, R|q)$, which is defined as a conditional probability distribution of a corrected word c and a rule set for the correction R conditioned on the misspelled word q. The model is specified as log linear model with substitution rules as features

$$P(c, R(c,q)|q) = \frac{\exp\{\sum_{r \in R(c,q)} \lambda_r\}}{\sum_{c'} \exp\{\sum_{r' \in R(c',q)} \lambda_{r'}\}},$$

where c denotes a correct word, q a misspelled word, $R(c,q)$ the set of rules of transforming c to q, r a substitution rule, and λ_r the weight of

rule r. In learning, contextual substitution rules are first derived from training data as in Figure 3.1, and then the parameters of the model are learned with a quasi-Newton method. In prediction, given a query word, the top k candidate corrections with respect to the model are guaranteed to be found, using a trie for storing the dictionary and a tire for storing the rule set, as well as an algorithm based on dynamic programming.

So far we have been considering correction of individual misspelled words. The accuracy of correction can be further improved by considering the surrounding words (context words). For example, one can decide whether to correct "officier" as "officer" or "official" by looking at the context words. If the next word is "website", then the correct word is more likely to be "official". Methods of using context words for spelling error correction have also been proposed [74, 118]. In spelling error correction, it is necessary that the right corrections of misspelled words are among the candidates. Methods for automatically collecting and utilizing such data from search results as well as web n-grams have also been proposed [43, 92].

3.2.2 Query Segmentation

Query segmentation is to separate the input query into multiple segments, roughly corresponding to natural language phrases, for improving search relevance. For example, if the query is "new york times square", then it may be divided into two phrases "(new york)(times square)", and the relevance may be enhanced by the segmentation. Both supervised and unsupervised learning methods have been proposed for query segmentation.

Bergsma and Wang [23], for example, propose a supervised method, which exploits a classifier and features such as position, part-of-speech tag, phrase frequency, and frequency of adjacent words. A segmentation of query $q = q_1 q_2 \cdots q_n$ of length n can be represented as $b = b_1 b_2 \cdots b_{(n-1)}$, where $b_i \in \{1, 0\}, i = 1, \cdots, n-1$ denotes a break decision between query words q_i and q_{i+1}, 1 stands for making a break, and 0 stands for not making a break. There are $n-1$ break decisions for

the query. Their classifier is a binary classification model that makes break/not-break judgments at the $n-1$ positions.

Bendersky et al. [16] suggest jointly performing query segmentation, capitalization, and POS tagging on query using conditional random fields (CRF). The approach takes advantage of the fact that the annotations of query segmentation, capitalization, POS tagging are interdependent of one another. For example, if a query word is likely to be a preposition, then it is not likely to be capitalized. The method first conducts the annotations independently, and then takes the initial annotations as input and makes prediction on the final annotations using the CRF model.

Hagen et al. and Tan & Peng propose unsupervised methods [168, 81, 80] for query segmentation, which make use of heuristic functions, wikipedia titles, and web n-grams. (See also [121]). The method in [80], referred to as Wekipedia-based Normalization (WBN), assigns a weight to each segment and sums up all the weights as the score of the entire segmentation. It then chooses the segmentations with the highest k scores. The score of segmentation S is defined as follow

$$score(S) = \begin{cases} \sum_{s \in S, |s| \geq 2} weight(s) & \text{if } weight(s) > 0 \text{ for all } s \in S \text{ and } |s| \geq 2 \\ -1 & \text{else} \end{cases}$$

where s is a segment and segments with length one are ignored. The weight of segment s is defined as follow

$$weight(s) = \begin{cases} |s|^2 + |s| \cdot \max_{t \in s, |t|=2} freq(t) & \text{if } s \text{ is Wikipedia title} \\ |s| \cdot freq(s) & \text{else} \end{cases}$$

where t denotes a substring of s and $freq$ denotes the frequency of string in the corpus.

Recently, Wu et al. [186] employ a re-ranking approach for query segmentation, which is a technique widely utilized for structure prediction in natural language processing. In the first stage, they adopt the method of WBN [80] to find the top k candidates. In the second

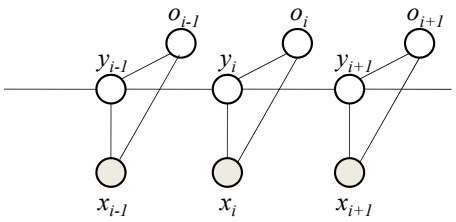

Figure 3.3: Conditional random fields for query reformulation.

stage, they create a feature vector for each candidate, employ SVM to re-rank the candidates, and then find the best segmentation as output. Features of the SVM model include mutual information between words in the segmentation, similarity between the current segmentation and the top ranked segmentation, and so on. The advantage of the re-ranking approach is that it can leverage the strengths of both the generative approach and the discriminative approach, i.e., to quickly identify a set of reasonably good candidates and to accurately select the best candidate using as much information as possible.

Other types of query reformulation have also been investigated. For example, query reduction has been studied in [10] and query stemming has been studied in [146].

3.2.3 General Query Reformulation

Here we consider methods that can address different types of query reformulation in a unified framework.

A discriminative method for general query reformulation has been developed by Guo et al. using conditional random fields (CRFs) [79]. The basic idea is as follows. We could in principle view query reformulation as a mapping from the space of original queries X to the space of reformulated queries Y. Obviously, directly exploiting the model $P(y|x)$ would be intractable, because both X and Y are extremely large, where y and x are random variables taking values from Y and X. Guo et al. propose adding another random variable o and employ the model $P(y, o|x)$ to solve the problem, where o takes values from a set of operation sequences. An operation can be insertion,

deletion, or substitution of letter in a word, splitting of a word into multiple words, merging of multiple words into a single word, word stemming, etc. The number of mappings from x to y under o will be drastically reduced. They define $P(y, o|x)$ as CRFs on query word sequences (Figure 3.3)

$$P(y, o|x) = \frac{1}{Z} \prod_{i=1}^{n} \phi(y_{i-1}, y_i)\phi(y_i, o_i, x),$$

where x denotes a sequence of query words, y a sequence of refined query words, o a sequence of operations, and ϕ a feature function. Features $\phi(y_i, o_i, x)$ can be those indicating word y_i is obtained from word x_i by operation o_i, where o_i is insertion, deletion, or replacement of a letter, and features $\phi(y_{i-1}, y_i)$ can be those indicating whether word y_{i-1} and word y_i co-occur in a corpus. Note that the model can be extended to handle multiple steps of query reformulation. They derive methods for learning the model and making prediction using dynamic programming. One advantage of this approach is that different types of query reformulation can be performed in the same framework and thus the accuracy of the task can be enhanced, because reformulations can be interdependent.

Li et al. [120] propose a generative approach to the task. A generalized hidden Markov model is employed, which can handle both in-word spelling errors (insertion, deletion, substitution, and misuse) and cross-word errors (splitting and concatenation) in a unified framework. In the model, the hidden states represent the correct forms of words associated with error types and the observations are (potentially) misspelled words. A Perceptron-based discriminative training method is used to train the parameters of the model.

3.3 Methods of Similar Query Mining

We describe methods of automatically mining similar queries or similar query patterns from search log data, including click-through data and session data, as well as web data (text data). (For a survey on search log mining, see [163, 94].) The mined similar queries or similar query patterns can be used in candidate generation of query reformulation.

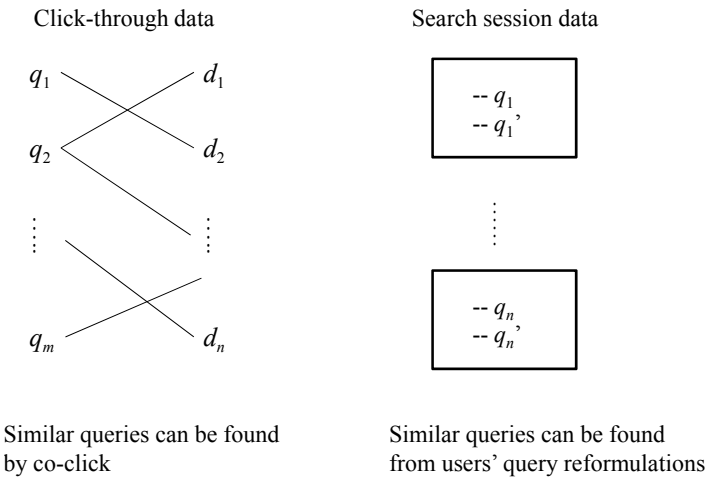

Figure 3.4: Search log data.

The challenge here is how to deal with noise in the data. We also introduce methods for automatically learning query similarities. A public dataset of similar queries is also introduced.

3.3.1 Using Click-through Data

Click-through data records the queries submitted at the search engine and the URLs clicked by the users at the search engine. Click-through data can be represented by a weighted bipartite graph, as shown in Figure 3.4, in which one set of nodes represent queries, the other set of nodes represent URLs, edges represent clicks between queries and URLs, and weights on the edges represent numbers of clicks.

In a click-through bipartite graph, if two queries have similar sets of URLs associated, then the queries tend to have similar meanings (represent similar search needs). One can find similar queries from a click-through bipartite graph on the basis of URLs associated with them.

One simple method is to use statistical measures such as Pearson correlation coefficient to characterize the similarities between two

queries [2].

$$r(q,p) = \frac{\sum_{i=1}^{n}(q_i - \bar{q})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{n}(q_i - \bar{q})^2}\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}},$$

where q and p denote query vectors, q_i and p_i denote the i-th elements of the vectors representing click frequencies, and $\bar{q}$ and $\bar{p}$ denote average frequencies. Intuitively, if two queries have many shared clicked URLs, then the coefficient score between the two queries will be large. Xu and Xu [191] find that when Pearson coefficient is larger than 0.8, more than 82.1% of query pairs may be viewed as similar query pairs.

Another approach to finding similar queries is to cluster queries on the basis of their clicked URLs on a click-through graph. For example, Beeferman et al. [12] propose employing an agglomerative clustering algorithm for performing the task. Their method represents queries with vectors of their clicked URLs, and iteratively merges the two queries or query clusters into a new cluster, if their similarity is the largest. Although the algorithm is simple, it can discover high quality clusters of similar queries. Other algorithms are also applied to query clustering, such as K-means [5] and DBSCAN [185].

One issue with the clustering approach is to scale up to large datasets. Cao et al. propose an efficient agglomerative clustering algorithm to achieve the goal, by leveraging the fact that a click-through graph is usually sparse (each node has a small number of edges associated) [39, 123]. The algorithm only scans the data once (with linear order time complexity) and incrementally creates clusters. More specifically, the algorithm creates and maintains an inverted index about all the non-zero elements of the query vectors in the existing clusters. Once a new query vector comes, it only takes its non-zero elements, looks up the index, and makes similarity comparison with the clusters that also have non-zero elements at the same positions.

Another state-of-the-art method for finding similar queries from a click-through graph is to perform random walk on the graph, proposed by Craswell and Szummer [50]. Their method takes the click-through bipartite graph as a directed graph and defines a random walk model

[2] Interestingly, Pearson correlation coefficient is also widely utilized in collaborative filtering as similarity measure between items.

on the graph. First, the transition probability matrix $A[i,j] = P(i|j)$ is defined as follows,

$$P(i|j) = \begin{cases} (1-s) \cdot \frac{f(i,j)}{\sum_k f(i,k)}, & \text{if } i \neq j \\ s, & \text{if } i = j \end{cases},$$

where i and j denote nodes i and j on the graph, $f(i,j)$ denotes the frequency of edge between nodes i and j, and s is the self-transition probability. Then, random walk (both forward and backward) can be performed on the graph. Given a test node k (say, representing a query), the input vector is constructed $v(0) = e_k$, where e_k is a unit probability vector with the k-th element being 1 and the other elements being 0. The backward random walk, for example, is realized by computing $v(n) = A \cdot v(n-1)$ where n denotes number of iterations. After several rounds of iterations, the nodes of similar queries will have higher weights. Their model is actually a model of similarity weight propagation on the click-through bipartite graph. The use of self-transition probability s is to ensure the propagation also goes to the node itself and does not go to the other nodes too much. Another method of finding similar queries is to calculate the hitting time of queries on the click-through graph, proposed by Mei et al. [133].

3.3.2 Using Session Data

Each instance of session data records the queries submitted and URLs clicked by the user in a search session (cf., Figure 3.4). How to segment log data into sessions is still an active research topic and one simple yet effective method is the so-called 30-minute rule, which regards any time interval longer than 30 minutes as a session boundary [41]. For the task of similar query mining, we usually only utilize the query sequences in the segmented sessions.

Users sometimes issue multiple similar queries in the same search sessions, and they can be synonymous, more general, or more specific expressions. We can also mine frequent successive query sequences from search sessions and take them as similar queries (cf.,[27]).

Jones et al. [96] propose finding frequent query pairs from search sessions by statistical hypothesis testing. Given two queries, they

calculate the likelihood ratio between them and perform likelihood ratio testing to see whether their co-occurrence in a search session is statistically significant. More specifically, suppose that the two queries are q_1 and q_2. We calculate the likelihood that q_2 occurs after q_1 does not occur as $L(q_2|\neg q_1)$ and the likelihood that q_2 occurs after q_1 occurs as $L(q_2|q_1)$, where q_1 and q_2 are assumed to be generated according to binominal distributions. We then calculate the ratio $-2\log L(q_2|\neg q_1)/L(q_2|q_1)$. If it is larger than a threshold, then we judge that the occurrence of q_2 is statistically dependent on the occurrence of q_1, i.e., we view q_1 and q_2 as similar queries. Other measures including cosine similarity and Jacaard similarity can also be used [89].

Xue et al. [197] propose mining question reformulation patterns (similar question patterns) from session data. They employ sequence pattern mining techniques to automatically mine 5w1h (who, what, where, when, why, and how) question reformulation patterns, which represent alternative expressions for 5w1h questions. For example, users may ask similar questions "how far is it from X to Y" and "distance from X to Y" in search sessions, where X and Y represent location names. They show that one can really mine high-quality reformulation patterns from a large amount of session data with this mining method. See also [182], for a method of mining and utilizing term association patterns in query log.

3.3.3 Using Document Collection

An alternative approach is to mine synonyms and synonymous patterns from large amounts of text data. For example, Turney [173] presents a method for mining synonyms from a large document collection based on the assumption that a word is characterized by the contexts it has. Given a word and candidates of its synonym, his method leverages a search engine to collect co-occurrence information, calculates mutual information between the word and each of the candidates, and selects as synonym the candidate having the largest mutual information. Lin and Pantel [125] propose a method of extracting synonymous patterns such as "X is author of Y" is equivalent to "X wrote Y", by parsing a large number of texts and mining similar paths of parsed trees in

Table 3.2: Similar queries to original TREC topic "Obama family tree".

barack obama family	obama family
obama s family	barack obama family tree
the obama family	barack obama s family
obamas	obama genealogy
barack obama s family tree	barack obama ancestry
president obama s family	obamas family
obama family history	obama s family tree
barack obama genealogy	barack obama family history
barack obama geneology	president obama and family
obama s ancestry	barak obama family tree
barak obama family	obama family tre
obama and family tree	

similar contexts. How useful such mined synonyms and synonymous patterns are needs more investigation.

3.3.4 Learning Query Similarity

Bona et al. [57] have developed a method for learning and predicting query similarity. The basic idea is to use click-through data and a learning to rank method to automatically train a model which can rank queries based on their similarity to a given query. Pairwise training data is derived from click-through log under the assumption that two queries are more similar if they are more frequently associated with the same documents. The features of the ranking model include mutual information between queries as in [96], edit distance between queries, and an extended edit distance between queries which takes semantic similarity into consideration. See also [191] for a method of learning query similarity in a metric space with similar query pairs as training data.

3.3.5 Public Dataset

Finally, we introduce a public dataset for research on query reformulation, called QRU-1 [3]. The QRU-1 dataset includes the topics in the TREC Web Track as original queries. It further contains approximately twenty queries similar to each of the original queries [117]. Table 3.2 gives an example. We can see that the similar queries really represent the same search need as the original query, but in different forms, such as synonyms, stemming variations, spelling errors, and abbreviations. The similar queries are automatically generated from a model [183] trained from search log data with the TREC topics as input. In addition, manual cleaning of the generated queries is performed to ensure only 'true' similar queries are retained in the dataset.

Li et al. [117] have conducted an experiment on relevance ranking using the QRU-1 dataset. They find that for a large number of queries there exists at least one reformulated query which can give better performance in relevance ranking. It is usually not clear, however, which one is the query. This poses an interesting question for future research.

3.4 Methods of Search Result Blending

We next describe methods of search result blending. Suppose that the original query is reformulated into similar queries. Then documents can be retrieved with the original query and similar queries. (To improve efficiency, we can have the search system support the mechanism that the queries are combined into one query representation and documents are retrieved at the same time with the query representation (multiple queries).) Relevance scores are calculated for and assigned to the documents with respect to each of the queries. The question then is how to 'blend' the search results of the queries. Note that the relevance scores of documents are not comparable across queries.

A simple method of blending is to employ a linear combination (cf.,[66, 112, 197]) with query similarities as weights. The final relevance

[3] The data is available at
http://research.microsoft.com/en-us/downloads/d6e8c8f2-721f-4222-81fa-4251b6c33752/

score $f(q, d)$ of document d with respect to query q is calculated as

$$f(q, d) = f_B(q, d) + \sum_i k(q, q_i) f_B(q_i, d),$$

where $f_B(q, d)$ and $f_B(q_i, d)$ denote the relevance scores of document d with respect to queries q and q_i calculated by the basic model, for example, BM25, and $k(q, q_i)$ denotes the similarity between query q and query q_i.

Learning to rank techniques can also be employed in search result blending. LambdaMerge is an algorithm designed for learning the matching and ranking models at the same time [160]. It employs LambdaRank as the learning algorithm and utilizes as the features query document matching scores, quality scores of query reformations, and quality scores of search results. The model of LambdaMerge is defined as follows.

$$f(q, d) = \sum_i w(q_i) f_B(q_i, d),$$

where $f(q, d)$ denotes the final relevance score of document d with respect to query q, $f_B(q_i, d)$ denotes the relevance score of document d with respect to query q_i by the basic model, and $w(q_i)$ denotes the weight with respect to query q_i. Note that queries q_i include the similar queries as well as the original query. Furthermore, the basic model $f_B(q_i, d)$ is calculated as

$$f_B(q_i, d) = f_N(x_i; \alpha),$$

where $f_N(\cdot; \cdot)$ is a two layer neural network, x_i denotes the feature vector of query q_i and document d, and α denotes parameters. The weight $w(q_i)$ is calculated as

$$w(q_i) = f_G(z_i; \beta),$$

where $f_G(\cdot; \cdot)$ is the softmax function, z_i denotes the feature vector with respect to query q_i, and β denotes parameters. The features in z_i represent the quality score of q_i and the quality score of search result with respect to q_i. Figure 3.5 depicts the LambdaMerge model.

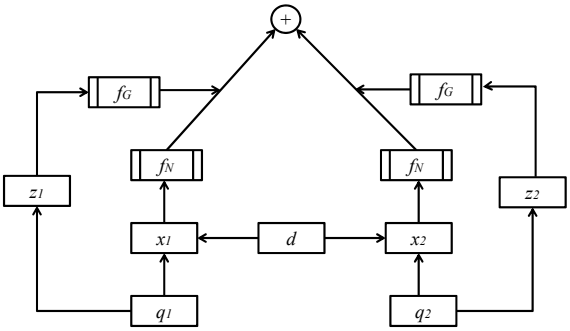

Figure 3.5: LambdaMerge model.

Another theoretically sound approach is to employ a kernel method, developed by Wu et al. [189, 194]. The blending model is defined as

$$f(q,d) = f_B(q,d) \times \sum_i \alpha_i k_Q(q,q_i) k_D(d,d_i) f_B(q_i,d_i),$$

where $k_Q(q,q_i)$ denotes the similarity between original query q and similar query q_i, $k_D(d,d_i)$ denotes the similarity between original document d and similar document d_i, α_i denotes the weight of query document pair (q_i,d_i), and $f_B(q,d)$ and $f_B(q_i,d_i)$ are basic relevance functions. Here, it is assumed that $0 \leq k_D(\cdot,\cdot) \leq 1$, $0 \leq k_Q(\cdot,\cdot) \leq 1$, and $f_B(\cdot,\cdot) > 0$ [4]. The weights can be automatically learned from click-through data by the kernel approach, such as Ranking SVM [86]. Note that multiplication instead of addition is used in the model in order to make the problem solvable by a kernel method. Wu et al. shows that if the basic model is BM25, then a more reliable and robust model can be learned by the method, which they call Robust BM25.

Figure 3.6 gives an intuitive explanation on why the use of the model can effectively deal with query document mismatch. Suppose that there exists a query space Q and similarity function k_Q is defined on it. Given query q, one can find its similar queries q_i in Q on the basis of $k_Q(q,q_i)$. Similarly, there exists a document space D and similarity function k_D is defined on it. Given document d, one can

[4] We can always add a small value ϵ to make $f_B(\cdot,\cdot) > 0$.

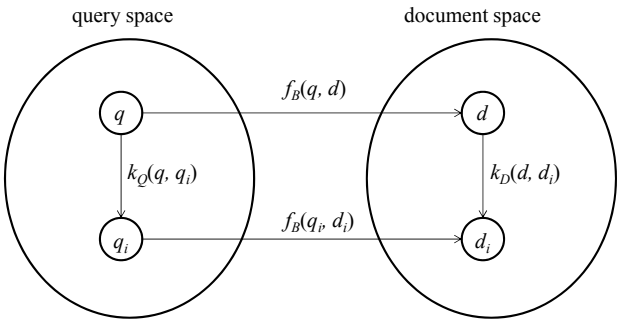

Figure 3.6: Kernel method for search result blending.

find its similar documents d_i in D on the basis of $k_D(d, d_i)$. The basic relevance model (matching model) $f_B(q, d)$ is defined between query q and document d over the two spaces. Query document mismatch occurs, when $f_B(q, d)$ cannot be reliably calculated. We can deal with the problem by employing the k-nearest neighbor method over the query and document spaces. More specifically, we smooth the basic relevance score of query q and document d by using the relevance scores of their similar queries q_i and similar documents d_i.

3.5 Methods of Query Expansion

Query expansion is a technique studied intensively and widely in IR. The basic idea is to enrich the query with additional terms (words or phrases) and to use the expanded query to conduct search in order to circumvent the query-document mismatch challenge. There are two major characteristics for the existing methods of query expansion. Namely, extra terms are added to the query and traditional IR models are utilized. More specifically, a traditional IR model can be written as

$$f(q, d) = \sum_{t \in q \cap d} w_{q,t} \cdot w_{d,t},$$

where q denotes a query, d denotes a document, t denotes a term, $w_{q,t}$ denotes the weight of term t in query q, and $w_{d,t}$ denotes the weight of term t in document d (see Section 7.1.2 for related discussions). The

model based on query expansion is written as

$$f(q', d) = \sum_{t \in q' \cap d} w'_{q',t} \cdot w_{d,t},$$

where q' denotes an expanded query and w' denotes a new weight.

One representative method of query expansion is pseudo-relevance feedback [147, 174, 192, 38]. It first conducts search with the original query, then automatically selects some terms from the returned documents, and finally performs another search with the expanded query (original query plus additional terms). The returned documents in the first search are assumed to be 'pseudo'-relevance feedback from the user.

We refer to the reader to the survey paper on query expansion by Carpineto and Romano [40]. Here, we introduce several recent works which attempt to perform query expansion by leveraging external resources. The approach is promising, because a large amount of knowledge and information is now available outside of the document collection.

In the work on blog search by Arguello et al. [4], query expansion is also carried out for solving the mismatch problem. The anchor texts within Wikipedia articles are used as resources for query expansion. Given a query, their method first retrieves Wikipedia articles with a conventional retrieval model. The method then adds an anchor text to the candidate set, if it appears in the top W retrieved Wikipedia articles and points to the top R retrieved Wikipedia articles ($R \leq W$). It then ranks the anchor texts in the candidate set with a heuristic function and selects the top 20 anchor texts for expanding the original query. Li et al. [122] also make use of Wikipedia articles for query expansion, in which terms are selected from the top ranked articles.

Kotov and Zhai [104] conduct query expansion with ConceptNet [5] [127], a graph-based knowledge base of common sense. In their method, given a query, the concept nodes on the graph matching the query terms and their neighbors are used to form a sub-graph. Heuristic methods as well as a learning-based method

[5] http://conceptnet5.media.mit.edu/

Table 3.3: Performances of query reformulation in search.

		MAP	NDCG@1	NDCG@2	NDCG@5
web	BM25	0.0908	0.1728	0.2019	0.2180
search	query expansion	0.0963	0.1797	0.2061	0.2237
	Robust BM25	**0.1192**	**0.2480**	**0.2587**	**0.2716**
enterprise	BM25	0.2745	0.4246	0.4531	0.4741
search	query expansion	0.2755	0.4076	0.4712	0.4958
	Robust BM25	**0.3122**	**0.4780**	**0.5065**	**0.5295**

(linear regression) are then employed to select the best concept nodes in the sub-graph. The selected concepts are used for expanding the query. Experimental results show that the learning-based method achieves the best performance.

Diaz and Metzler [60] present a formal method for query expansion with external corpora. The relevance model in [109] is generalized to combine evidence from both the document collection and external collections. They first conduct retrieval with the original query on each collection and build a relevance model with the top ranked documents. They then create a linear combination of the models on all the collections, as shown below.

$$P(w|\theta_q) = \sum_{c \in C} P(c)P(w|\theta_q, c),$$

where C is a set of collections and $P(w|\theta_q, c)$ is the relevance model of collection c. Their experimental results show that expansion on a suitable external collection is more effective than expansion on the original collection. Bendersky et al. [18] propose a unified framework for automatically optimizing the combination of multiple information sources for query reformulation. The framework supports arbitrary query fragments (e.g., unigrams, bigrams, expanded terms), weighting of query fragments, and automatic tuning of the model.

3.6 Experimental Results

We present the experimental results of improving search relevance by query reformulation, reported in [189]. In the experiment, a dataset of web search and a dataset of enterprise search are utilized, and mean average precision (MAP) and normalized discounted cumulative gain (NDCG) [93] are adopted as evaluation measures. The results in Table 3.3 indicate that query reformulation with suitable blending and mining (the Robust BM25 method) can significantly outperform the baselines of BM25 and query expansion. All the improvements of Robust BM25 over the baselines are significant (t-test, $p < 0.05$).

4

Matching with Term Dependency Model

We can address the mismatch challenge by exploiting a model based on multiple terms (phrases, n-grams, etc.), referred to as term dependency model (or dependency model). This is because concepts (word senses) are often represented by multiple words in English, such as "United States", and the relevance between query and document can be better characterized by matching based on multiple terms. In this section, we first give an overview on using term dependency in search and then describe several methods of leveraging term dependency, including Markov random fields (MRF) model, extended IR model, and features of learning to rank. We also give experimental results to demonstrate how well term dependency models can perform favorably against the traditional model of BM25.

4.1 Term Dependency

Search is basically performed by matching between query and document, without 'understanding' of the meaning of them. Within this matching framework, one rule of thumb is that matching of consecutive terms between the query and document indicates stronger relevance.

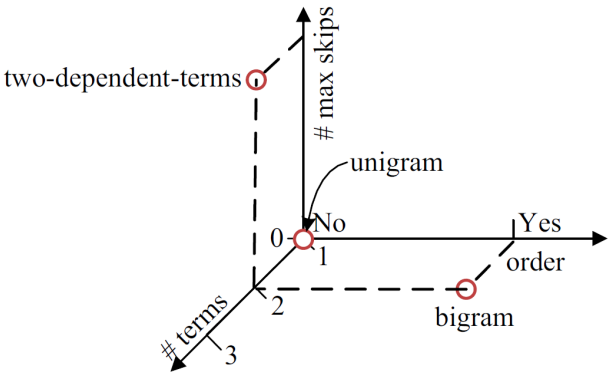

Figure 4.1: Three factors of term dependency.

For example, if terms "hot" and "dog" occur together in the query, then the document in which the two terms also occur together is more relevant than the document in which it is not the case. Another rule is that the order of terms in a query is quite free, but not completely free, and matching of ordered terms between the query and document indicates stronger relevance. For example, "hot dog recipe" represents the same search need as "recipe hot dog", but not as "hot recipe dog". Therefore, we need to consider the proximity as well as the order of terms in both query and document, when conducting matching between query and document in search.

Proximity and order can be realized by using n-grams of terms, in other words, term dependency or term collocation, which represent soft segmentation of query and document. There are several factors on which we can characterize n-grams of terms.

1. Number of terms: the number of terms in n-gram

2. Order: the order of terms is free or not

3. Skip: the maximum number of terms skipped within n-gram

Different choices of the factors lead to different types of term dependencies, as shown in Figure 4.1. For example, bigram means two consecutive words without skipping a word between them.

For example, when the query is "hot dog recipe", the bigrams of the query become "# hot", "hot dog", "dog recipe", where # denotes the start symbol. Bigrams in the document can be defined similarly. Matching between the n-grams in the query and document can be conducted, and relevance between the query and document can be calculated.

There are two approaches to using term dependencies. One approach is to combine all the matching scores of n-grams in a single model. The other approach is to take the matching scores of n-grams as features in the ranking model and employ learning to rank techniques to train the ranking model. The MRF model and extended IR model belong to the first approach.

4.2 Methods of Matching with Term Dependency

4.2.1 MRF Model

The MRF models can be categorized into two groups based on two types of dependency: explicit term dependency and latent term dependency. MRF models of the former group directly leverage the dependencies of terms in the query and document in matching. MRF models of the latter group make use of the dependencies of terms not only in the query and document, but also in other resources such as expanded query and Wikipedia.

MRF is a joint probability distribution represented by an undirected graph in which a node denotes a random variable and an edge denotes probabilistic dependency. Figure 4.2 shows an MRF with five nodes and five edges.

The joint probability of MRF can be factorized into the product of potential functions defined on the cliques of the graph.

$$P(x_1, \cdots, x_N) = \frac{1}{Z} \prod_{c \in \text{clique(G)}} \psi(c),$$

where c denotes a clique of graph, ψ denotes a potential function defined on a clique, and Z denotes the normalization function. A clique is a maximal complete subgraph.

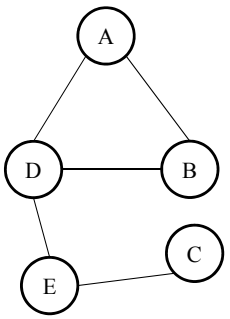

Figure 4.2: Example of Markov random fields.

Metzler and Croft [136] propose an MRF model for combining matching scores (relevance scores) of dependent terms. In MRF, the document is represented as a node and each term in the query is also represented as a node. The document node is connected to all the query term nodes, while the edges represent the matching relations between the query terms and the document. Dependent terms are also connected with each other, while the edges represent the dependencies between the query terms. The term dependencies are determined in advance, on the basis of certain assumptions such as bigram assumption and name-entity assumption.

Figure 4.3 shows three types of dependency. The left figure shows full term independency which means that the query terms are mutually independent given the document, i.e., $P(q_i, q_j|d) = P(q_i|d)P(q_j|d)$. Conventional models such as VSM, BM25, and LM4IR make the assumption. The middle figure shows sequential term dependency which means the adjacent query terms are mutually dependent, i.e., $P(q_1, \cdots, q_n|d) = \prod_{i=1}^{n} P(q_i|q_{i-1}, d)$. The right figure shows full term dependency which means all query terms are dependent with each other.

In MRF, the joint probability of query q and document d can be formally represented as

$$P(q, d) = \frac{1}{Z} \prod_{c \in \text{clique(G)}} \exp(\lambda_c f(c)),$$

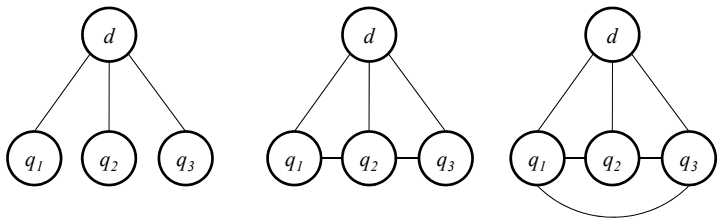

Figure 4.3: Three types of dependency.

where c denotes a clique, $f(c)$ denotes a feature function, and λ_c denotes a weight. We consider using the following matching function, which is rank equivalent to the joint probability function

$$F(q,d) = \sum_{c \in \text{clique(G)}} \lambda_c f(c). \tag{4.1}$$

Here, rank equivalence means that the ranking lists created based on the scores from the two functions are exactly the same.

In [136], three types of feature functions are defined to capture the three types of term dependency. The first type of feature function corresponds to full term independency, defined as

$$f_1(q_i,d) = \log\left[(1-\alpha)\frac{tf(q_i,d)}{|d|} + \alpha\frac{cf(q_i)}{|C|}\right],$$

where α is a smoothing factor, $tf(q_i,d)$ is the frequency of q_i in d, $cf(q_i)$ is the frequency of q_i in the whole collection C, and $|\cdot|$ is the length of document or collection. The second type of feature function corresponds to sequential term dependency, defined as

$$f_2(q_i,\cdots,q_{i+k},d) = \log\left[(1-\alpha)\frac{tf(q_i,\cdots,q_{i+k},d)}{|d|} + \alpha\frac{cf(q_i,\cdots,q_{i+k})}{|C|}\right],$$

where $tf(q_i,\cdots,q_{i+k},d)$ denotes the number of times the ordered terms $(q_i,\cdots,q_{i+k})$ occur in document d, $cf(q_i,\cdots,q_{i+k})$ denotes the number of times the ordered terms occur in the whole collection. The third type of feature function corresponds to full term dependency, defined as

$$f_3(q_i,\cdots,q_j,d) = \log\left[(1-\alpha)\frac{tf(q_i,\cdots,q_j,d)}{|d|} + \alpha\frac{cf(q_i,\cdots,q_j)}{|C|}\right],$$

where $tf(q_i, \cdots, q_j, d)$ denotes the number of times the unordered terms $(q_i, \cdots, q_j)$ appear within a window of N terms in document d, and $cf(q_i, \cdots, q_j)$ denotes the number of times the ordered terms appear in the whole collection.

Metzler and Croft also propose a heuristic method to tune the parameters of the model (4.1) by directly maximizing the mean average precision (MAP) of the training data.

One extension to the basic MRF model is to include query expansion, called latent concept expansion [137]. In the model, the term nodes represent not only terms in the original query, but also terms from query expansion. In the original MRF model, all terms have an equal weight. Bendersky et al. [17] propose assigning different weights to different terms. The parameters in the weighted dependence model is then trained using learning to rank techniques. Lang et al. [108] identify hierarchical structures of documents and incorporate the hierarchical structures into term dependencies. Lease [110] proposes an improved MRF model for supporting verbose queries. In [15], Bendersky and Croft suggest using a hypergraph to represent higher order dependencies, i.e., dependencies between query concepts rather than query terms. See also [162, 132].

MRF models need to use the statistics of dependent terms. Thus, substantial time or space overhead will occur, when the conventional inverted index is used. Huston et al. [91] have developed a new type of index structure for efficient calculation of term dependency models. The key ingredient of the index is to exploit data stream sketching techniques to estimate n-gram statistics, which can minimize space usage and retain high accuracy of estimation.

4.2.2 Extended IR Model

There is a heuristic way of defining term dependency models, by extending BM25 and LM4IR defined on terms (unigrams) to models based on n-grams. Xu et al. [193] have investigated the approach and refer to the extended models as asymmetric kernels. For example, the

BM25 kernel is defined as follows.

$$\text{BM25-Kernel}(q,d) = \sum_t \text{BM25-Kernel}_t(q,d),$$

where $\text{BM25-Kernel}_t(q,d)$ denotes the BM25 kernel of type t.

$$\text{BM25-Kernel}_t(q,d) = \sum_x \text{IDF}_t(x) \times \frac{(k_3+1) \times f_t(x,q)}{k_3 + f_t(x,q)} \times \frac{(k_1+1) \times f_t(x,d)}{k_1 \left(1 - b + b\frac{f_t(d)}{avgf_t}\right) + f_t(x,d)},$$

where x denotes an n-gram of type t and t can be bigram, trigram, etc. Although it is simple, the approach works quite well. The extended IR models can always beat their counterparts, i.e., the basic IR models.

Gao et al. [70] propose incorporating dependency structure into LM4IR, called dependency language model. It is assumed that term dependencies in a query are represented by a linkage l: an acyclic, planar, and undirected graph where a node denotes a query term and an edge denotes the dependency between query terms. Query q is generated from document d through linkage l in two steps.

1. Linkage l is first generated according to conditional probability distribution $P(l|d)$.

2. Query q is then generated according to conditional probability distribution $P(q|l,d)$.

The conditional probability $P(q|d)$ is calculated over all possible linkages

$$P(q|d) = \sum_l P(q,l|d) = \sum_l P(l|d)P(q|l,d).$$

The authors exploit a heuristic method to estimate the parameters of the model.

Park et al. [145] have developed a term dependency model based on the quasi-synchronous stochastic process developed in natural language processing. A synchronous model generates the parsing tree of a target sentence by recursively matching the child nodes of a parent node of the parse tree of a source sentence. Four types of relations in a parse

tree are considered, such as, parent-child and ancestor-descendent. The model also adopts the framework of LM4IR and defines the conditional probability of generating a query from a document as the conditional probability of the parse tree of query T_q given the parse tree of document T_d through alignment A:

$$P(q|d) \approx P(T_q, A|T_d) = P(A|T_d)P(T_q|A, T_d).$$

Here alignment A denotes the set of possible matches between fragments of the syntactic trees of query and document. The authors also propose a method of estimating the parameters of the model.

4.2.3 Features of Learning to Rank

One can also define n-gram based features and proximity features, incorporate them into the ranking model, and employ learning to rank techniques to automatically train the weights of the features.

Tao and Zhai [170] have studied the effectiveness of a number of proximity features in relevance ranking. The proximity features include span, min coverage, minimum pair distance, average pair distance, and maximum pair distance, which measure how the n-grams in query match the document. For example, span is defined as the length of the shortest segment that covers all query term occurrences in the document. See also [166].

Svore et al. [167] have conducted comprehensive experiments to test the effectiveness of proximity features in the learning to rank framework. They have found that span-based features are particularly powerful for enhancing the performance of relevance ranking for different types of queries (long vs. short, head vs. tail).

Wang et al. [175] propose extracting key n-grams from web pages and making use of the extracted key n-grams to enrich the representations of the web pages. Specifically, 'search-focused' key n-grams are extracted, in the sense that they can compose 'good queries' for searching the pages. The extracted key n-grams are then used to define features in the learning to rank model, such as n-gram BM25. The key n-grams can capture the dependency information in

Table 4.1: Performances of term dependency models in search.

	fully independent		sequentially dependent		fully dependent	
	MAP	P@10	MAP	P@10	MAP	P@10
AP	0.1775	0.2912	0.1867*	0.2980	0.1866*	0.3068*
WSJ	0.2592	0.4327	0.2776*	0.4427	0.2738*	0.4413
WT10g	0.2032	0.2866	0.2167*	0.2948	0.2231*	0.3031
GOV2	0.2502	0.4837	0.2832*	0.5714*	0.2844*	0.5837*

queries and web pages well and thus help significantly improve relevance ranking, particularly for tail pages.

4.3 Experimental Results

We present the experimental results on using term dependency models for improving search relevance, reported in [136]. In the experiment, TREC collections (WSJ, AP, WT10g, and GOV2) and topic titles are used as documents and queries respectively. The evaluation measures are MAP and Precision at 10 (P@10). The results in Table 4.1 indicate that the sequentially dependent model and fully dependent model can outperform the baseline of fully independent model in terms of all measures. Some of the improvements are statistically significant, marked with '*'.

5

Matching with Translation Model

Statistical machine translation (SMT) refers to statistical learning techniques for translating natural language texts from one language to another language. Queries can be viewed as texts in one language and documents as texts in another language, and SMT technologies can be leveraged to deal with query document mismatch in search. More specifically, "ny" can match "New York" with a high degree of confidence, because the probability of translating the former to the latter is high. Matching by translation model in principle needs to learn the model from queries and associated documents, which can be obtained from click-through log, and thus belongs to the supervised learning approach to matching. This section first gives an introduction to SMT. It then describes how to perform matching using translation models in search. It also shows some experimental results obtained in previous work.

5.1 Statistical Machine Translation

SMT can be depicted with the source channel model. Given a sentence in the source language (e.g., in Chinese) $C = c_1 c_2 \cdots c_J$, we want to

find the best translation, the most suitable translated sentence in the target language (e.g., in English), $E = e_1 e_2 \cdots e_I$,

$$E^* = \arg\max_E P(E|C),$$

where arg max stands for an algorithm to find the target sentence with the highest probability among all possible ones. Applying Bayes' rule and discarding the constant denominator, we obtain

$$\begin{aligned} E^* &= \arg\max_E \frac{P(C|E)P(E)}{P(C)} \\ &= \arg\max_E P(C|E)P(E), \end{aligned}$$

where $P(E)$ denotes the language model for calculating the generation probability of E and $P(C|E)$ denotes the translation model for calculating the transformation probability from E to C.

Given an input sentence unseen in training data, an SMT system performs translation as follows. First, the input sentence is broken into small units; then, each unit is translated from source language to target language; finally the translated units are combined to form an output sentence. Different translation models differ in how the translation units are defined, translated, and combined. Existing translation models fall into three categories: word-based model [34], phrase-based model [142, 100, 46], and syntax-based model [198]. Word-based models are more frequently used for addressing the term mismatch problem in search.

Word-based models make use of words as basic translation units. In [34], a series of word-based translation models with increasing complexities have been proposed, called the IBM Models. IBM Model One is one of the most widely used word-based models. The target sentence $E = e_1 e_2 \cdots e_I$ is assumed to be generated from the source sentence $C = c_1 c_2 \cdots c_J$ by the translation model in the following way.

1. Choose the length of target sentence I, according to distribution $P(I|C)$
2. For each position $i (i = 1, 2, \cdots, I)$
   (a) Choose position j in the source sentence C according to $P(j|C)$

(b) Generate target word e_i according to $P(e_i|c_j)$

Therefore, the probability of target sentence E given source sentence C is calculated as follows.

$$\begin{aligned} P(E|C) &= \sum_a P(E,a|C) \\ &= \sum_{a(1)=0}^{J} \cdots \sum_{a(I)=0}^{J} P(E,a|C) \\ &= \sum_{a(1)=0}^{J} \cdots \sum_{a(I)=0}^{J} \frac{P(I|C)}{(J+1)^I} \prod_{i=1}^{I} P(e_i|c_{a(i)}) \\ &= \frac{P(I|C)}{(J+1)^I} \sum_{a(1)=0}^{J} \cdots \sum_{a(I)=0}^{J} \prod_{i=1}^{I} P(e_i|c_{a(i)}) \\ &= \frac{\epsilon}{(J+1)^I} \prod_{i=1}^{I} \sum_{j=0}^{J} P(e_i|c_j), \end{aligned}$$

where a is an alignment between words in the source and target sentences and $p(e_i|c_j)$ is the probability of translating word c_j to word e_i. In the model it is assumed that the choice of length I is independent of C and $P(I|C) = \epsilon$, a small constant. It is further assumed that all positions in the source sentence, including position zero for the null word, are equally likely to be chosen, yielding $P(j|C) = \frac{1}{J+1}$. Refer to Section 4.2.3 of [99] for the derivation of the last equation.

Figure 5.1 shows an example of translating a Chinese sentence to an English sentence. In the figure only the best alignment is shown. Given a training corpus $\{(C^{(k)}, E^{(k)})\}_{k=1}^{N}$, the parameters of the translation model can be estimated with the expectation maximization (EM) algorithm. More detailed explanations on SMT can be found in [99].

5.2 Search as Translation

5.2.1 Basic Model

Berger and Lafferty [22] propose formulating search as SMT problem, in which query q is translated into document d with the largest conditional

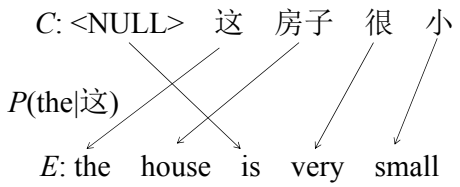

Figure 5.1: Translating Chinese sentence into English. Only one alignment is shown.

probability $P(d|q)$. One advantage of the approach is that it can deal with the query document mismatch problem. The model is written as

$$P(d|q) \propto P(q|d)P(d),$$

where $P(q|d)$ denotes a translation model translating d to q and $P(d)$ denotes a language model giving rise to d. Suppose that a query contains term "airplane" while the document only contains term "aircraft". There exists mismatch between the query and document. With the translation approach, however, it is very likely that the document term "aircraft" can be translated to the query term "airplane" with high probability and the term mismatch problem can be effectively solved. The translation model $P(q|d)$ can be estimated with queries and their associated documents (e.g., titles of documents) in click-through log, and the language model $P(d)$ can be estimated with documents [1].

Several issues need to be addressed, when applying SMT to search, including self-translation and training data.

5.2.2 Self-translation

One important difference between conventional machine translation and machine translation for search is that both queries (target language) and documents (source language) are in the same language. The probability of translating a word to itself should be quite high $P(w|w) > 0$, which corresponds to term matching in search. How to accurately calculate self-translation probabilities becomes an important

[1] The language model can also be learned with a different scheme, for example, BM25, as in cross-language information retrieval (CLIR) [177].

issue, therefore. If the self-translation probabilities are too large, then it will make the other translation probabilities small and decrease the effect of using translation. On the other hand, if the self-translation probabilities are too small, then it will make direct matching less effective and hurt the performance of matching.

A number of methods have been proposed to help estimate self-translation probabilities. All these methods assume that self-translation probabilities estimated directly from data are not optimal for the search task, and the authors have demonstrated that significant improvements can be achieved by adjusting the probabilities [68, 98]. One popular method is to linearly interpolate the probability of generating a query word from the document with the probability of translating the query word from the document, as explained below.

5.2.3 Training Data

The performance of statistical translation heavily relies on the data used for training. One of the key challenges in employing translation model in search is collection of high quality data. Ideally we would expect to use relevant query-document pairs (judged by human annotators) as training data. However, it is not practical to create sufficient amount of such data.

A number of methods have been developed for creating relevant query-document pairs. In [22], as the first study of using translation model for search, synthetic queries are generated to learn the translation probabilities. It is also observed that title-body pairs extracted from web pages can be taken as approximation of relevant query-document pairs. A large number of title-body pairs are utilized for estimating translation probabilities in the work by Jin et al. [95], referred to as title language model. In [98] Karimzadehgan and Zhai estimate translation probabilities in an unsupervised manner, based on normalized mutual information between words.

Click-through data in search naturally provides relevant query-document pairs. Figure 5.2 gives an example of URL (as well as its title) and its associated queries extracted from click-through data. Gao

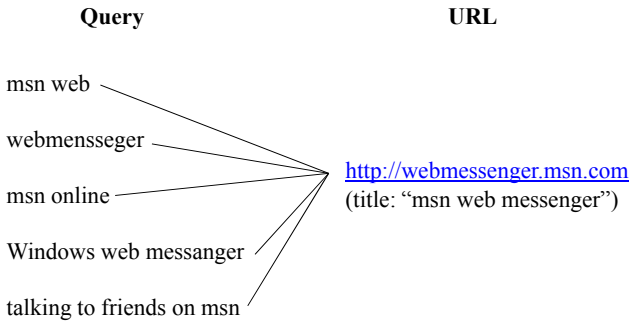

Figure 5.2: An example of URL and its associated queries extracted from click-through data.

et al. show that translation models learned from click-through log data can improve relevance ranking of web search [68].

Web pages have a number of fields (e.g., title, anchor, body, and query-click) and large differences exist among these fields in their nature. Analysis has been conducted to investigate which fields are suitable for being used as training data of translation model. Studies have demonstrated that translation models trained with title and query-click fields are very effective for improving relevance performance [68]. Click through data is not always available, however. Our experiences have shown that the use of title and anchor fields as training data is also an effective solution [2].

5.3 Methods of Matching with Translation

Berger and Lafferty [22] exploit word-based translation model for addressing the term mismatch problem. Given query q and document d, the word-based translation model calculates the translation probability

[2] Our unpublished work.

$P(q|d)$ in the following way.

$$\begin{aligned} P(q|d) &= \frac{P(m|d)}{(n+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{n} P(q_j|d_i) \\ &= P(m|d) \prod_{j=1}^{m} \sum_{i=0}^{n} \frac{1}{n+1} P(q_j|d_i) \\ &= P(m|d) \prod_{j=1}^{m} \left(\frac{n}{n+1} \sum_{i=1}^{n} \frac{1}{n} P(q_j|d_i) + \frac{1}{n+1} P(q_j|d_0) \right) \\ &= P(m|d) \prod_{j=1}^{m} \left(\frac{n}{n+1} P(q_j|d) + \frac{1}{n+1} P(q_j|\langle null \rangle) \right), \end{aligned}$$

where d_i is the i-th word of document d, m is the length of query q, n is the length of document d, $P(q_j|d) = \sum_{i=1}^{n} \frac{1}{n} P(q_j|d_i)$ is the probability of word q_j being translated from document d, $\langle null \rangle = d_0$ is the *null word* introduced in position zero of the document, and $P(q_j|\langle null \rangle)$ is the probability of word q_j being translated from the null word. $P(q_j|\langle null \rangle)$ plays the role of smoothing to avoid zero probability.

$P(q_j|d)$ can be further written as

$$P(q_j|d) = \sum_{w \in d} P(q_j|w) Q(w|d), \tag{5.1}$$

where $Q(w|d)$ is the *un-smoothed* probability of w being generated from document d by the document language model and $P(q_j|w)$ is the probability of word w being translated to word q_j by the translation model.

Gao et al. [68] show that the model still does not perform well in practical experiments due to low self-translation probabilities. Linear interpolation with language model is effective for coping with the problem. More specifically, the probability $P(q_j|d)$ in Equation (5.1) is adjusted as $P'(q_j|d)$

$$P'(q_j|d) = \beta Q(q_j|d) + (1-\beta) \sum_{w \in d} P(q_j|w) Q(w|d),$$

where $Q(q_j|d)$ is the un-smoothed probability by the document language model and β is the interpolation parameter.

Table 5.1: Performances of word-based translation models in search.

	NDCG@1	NDCG@3	NDCG@10
BM25 (baseline)	0.3181	0.3413	0.4045
WTM (without self-translation)	0.3210	0.3512	0.4211
WTM (with self-translation)	**0.3310**	**0.3566**	**0.4232**

Gao et al. [68] also propose using phrase-based translation model. The model represents the translation probability of a phrase in the query given a phrase in the title of a document. Phrase-based translation model is more powerful than word-based model because dependencies between words are considered in the model.

Statistical translation models have also been applied to query expansion. For example, Riezler et al. [150] suggest utilizing word-based translation model for query expansion. The model is trained with click-through data consisting of queries and snippets of clicked web pages. Gao and Nie [69] generalize word-based translation model to concept-based model and employ the model in query expansion. A concept can be an individual term, a contiguous phrase, or a pair of terms in proximity. The model is trained with click-through data containing queries and the titles of clicked web pages.

Statistical translation models have been applied to other tasks as well. Hillard et al. [87] addresses the term mismatch problem in sponsored search by using word-based translation model. Berger et al. [21] apply machine translation model to the task of finding answers to questions in question answering.

5.4 Experimental Results

We introduce the results on using translation models for improving search relevance, reported in [68]. The dataset contains 12,071 queries. Each query-document pair is assigned a relevance label at five level scale. Only the title fields of documents are used for training and evaluation. The evaluation measures are normalized discounted cumulative gain (NDCG) at the positions of 1, 3, and 10. The results

in Table 5.1 indicate that the word-based translation model (WTM) can outperform the baseline method of BM25 in terms of all measures. The performance of word-based translation model can be further improved by adding self-translation. All the improvements of WTM (with self-translation) and WTM (without self-translation) over BM25 are significant. WTM (with self-translation) significantly outperforms WTM (without self-translation) in terms of NDCG@1 and NDCG@3.

6

Matching with Topic Model

Query and document that are concerned with the same topics are likely to be relevant. For example, query "apple computer" shares the same topics with a document about "iPhone", but not with a document about "fruit". Therefore, it is likely that the query is relevant to the former document, not the latter document. If we can identify the topics of query and document and check whether they match each other, then we will be able to make reasonable judgment on the relevance between them. In this section, we first introduce topic modeling techniques, including probabilistic and non-probabilistic approaches. The models of probabilistic latent semantic indexing (PLSI), latent Dirichlet allocation (LDA), latent semantic indexing (LSI), non-negative matrix factorization (NMF), and regularized latent semantic indexing (RLSI) are explained in detail. We next describe how to use topic modeling techniques to deal with query document mismatch in search, including topic matching and smoothing. Finally, we introduce some experimental results reported in previous work.

Matching with topic model belongs to the unsupervised learning approach in the sense that the topic model is learned only from the

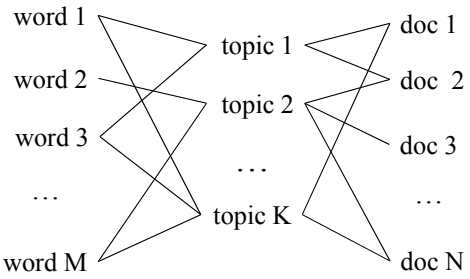

Figure 6.1: Topic model reveals relations between words and documents in a large document collection.

document collection. Query is regarded as a pseudo-document and query-document matching is carried out on topic aspect.

6.1 Topic Models

Given a collection of documents, topic modeling techniques aim to discover the topics in the collection as well as the topic representations of the documents. Basically, a topic model represents the relationship between the words and documents on the basis of *latent* topics, as shown in Figure 6.1. A topic is defined as a probability distribution over terms or a cluster of weighted terms. A document is defined as a set of words generated from a mixture of latent topics. Intuitively, words associated with a topic would more frequently appear together in documents, and thus they would be clustered into the topic. A document would talk more about certain topics, and thus the document would be mainly represented by the topics.

Various topic modeling methods, such as PLSI [88], LDA [26], LSI [58], NMF [111], and RLSI [180] have been proposed and successfully applied to different applications. In general, there exist two approaches to topic modeling: probabilistic approach and non-probabilistic approach.

6.1.1 Probabilistic Topic Models

Probabilistic topic models are usually generative models that can give rise to the documents in the collection. Suppose that $\mathcal{D} = \{d_1, d_2, \cdots, d_N\}$ is a set of documents with size N and $\mathcal{V}$ is a set of terms or words with size M, i.e., the vocabulary. A document $d \in \mathcal{D}$ consists of $|d|$ words from the vocabulary, denoted as $d = (w_1, w_2, \cdots, w_{|d|})$. Suppose that there are K topics in the document collection. In a probabilistic topic model, each topic is defined as a probabilistic distribution over terms in the vocabulary. Each document in the collection is defined as a probabilistic distribution over the topics.

PLSI [88] is one of the widely used probabilistic topic models. One can generate the documents in the collection in the following way.

1. select a document d from the collection with probability $P(d)$
2. for each document d in the collection
   (a) select a latent topic z with probability $P(z|d)$
   (b) generate a word w with probability $P(w|z)$

Here $z \in \{z_1, \cdots, z_K\}$ is a latent variable representing a topic. Figure 6.2 shows the graphical representation of the PLSI model.

The parameters of $P(d), P(w|z)$, and $P(z|d)$ can be estimated by maximizing the log-likelihood function with the expectation maximization (EM) algorithm.

$$L = \sum_{n=1}^{N} \sum_{m=1}^{M} f(d_n, w_m) \log P(d_n, w_m),$$

where $f(d_n, w_m)$ is the frequency of w_m in d_n. The joint probability of document-word pair $P(d, w)$ is calculated as

$$P(d, w) = P(d)P(w|d) = P(d) \sum_{z} P(w|z)P(z|d).$$

LDA [26] is another well-known probabilistic model. The main characteristics of it is that the topic distribution is assumed to have a Dirichlet prior, which will yield smoother estimates of probabilities. The generation procedure of LDA is as follows.

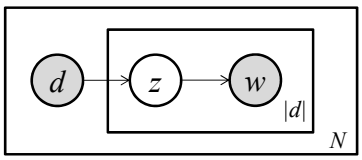

Figure 6.2: Graphical representation of probabilistic latent semantic indexing.

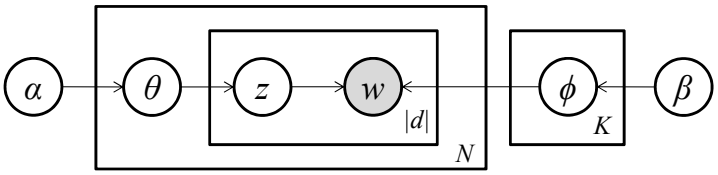

Figure 6.3: Graphical representation of latent Dirichlet allocation.

1. for each topic $k = 1, \cdots, K$
   (a) draw word distribution ϕ_k according to $\phi_k|\beta \sim Dir(\beta)$
2. for each document d in the collection
   (a) draw topic distribution θ according to $\theta|\alpha \sim Dir(\alpha)$
   (b) for each word w in the document d
      i. draw a topic z according to $z|\theta \sim Mult(\theta)$
      ii. draw a word w according to $w|z, \phi_{1:K} \sim Mult(\phi_z)$

The graphical representation of LDA is shown in Figure 6.3. The total probability of the model is

$$P(w, z, \phi, \theta; \alpha, \beta) = \prod_{k=1}^{K} P(\phi_k; \beta) \prod_{n=1}^{N} P(\theta_n; \alpha) \prod_{w \in d_n} P(z|\theta_n) P(w|\phi_z).$$

Gibbs sampling and variational EM can be employed for the parameter estimation and inference in LDA.

Probabilistic topic models have been employed in a number of text processing tasks. For example, Brants et al. [28] apply PLSI to document segmentation, i.e., identifying boundaries between parts

of document that have different topic distributions. Haghighi and Vanderwende [82] apply topic modeling techniques to multi-document summarization in which the topic model is utilized for representing the content of a set of documents. Krestel et al. [105] try to annotate tags to images using LDA. Specifically, they first identify topics of images from their associated tags with LDA and then assign the major tags of the topics to new images.

6.1.2 Non-probabilistic Topic Models

Non-probabilistic topic models are usually obtained by matrix factorization. Suppose that $\mathcal{D}$ is a set of documents with size N and $\mathcal{V}$ is a set of terms or words, i.e., a vocabulary with size M. A document is represented as an M-dimensional vector d with the m-th entry being the weight of the m-th term (e.g., tf-idf). The document collection $\mathcal{D}$ is represented as an $M \times N$ term-document matrix $\mathbf{D} = [d_1, \cdots, d_N]$, where the m-th row corresponds to the m-th term and the n-th column corresponds to the n-th document. A topic is also represented as an M-dimensional vector u, where the m-th entry denotes the weight of the m-th term in the topic. Terms having larger values are more indicative of the topic. Suppose that there are K topics. The K topics amount to an $M \times K$ term-topic matrix $\mathbf{U} = [u_1, \cdots, u_K]$, where the m-th row corresponds to the m-th term and the k-th column corresponds to the k-th topic.

Topic modeling here means discovering the latent topics in the document collection as well as representing the documents as mixtures of topics. More specifically, document d_n is succinctly represented as $d_n \approx \sum_{k=1}^{K} v_{kn} u_k = \mathbf{U} v_n$, where v_{kn} denotes the weight of the k-th topic u_k in the document. The larger value weight v_{kn} has, the more important role topic u_k plays in the document. The document representations amount to topic-document matrix $\mathbf{V} = [v_1, \cdots, v_N]$, where column v_n stands for the representation of document d_n in the latent topic space. That is to say, term-document matrix $\mathbf{D}$ is factorized by term-topic matrix $\mathbf{U}$ and topic-document matrix $\mathbf{V}$:

$$\mathbf{D} \approx \mathbf{U}\mathbf{V}^T.$$

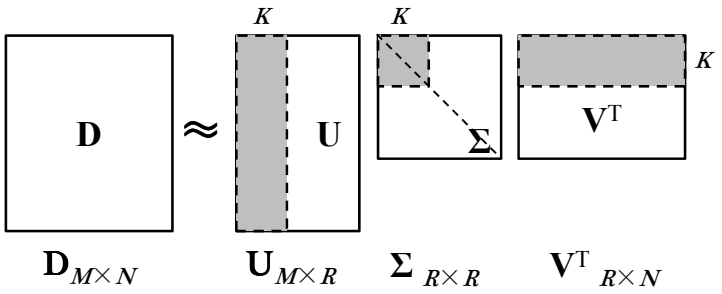

Figure 6.4: In latent semantic indexing, matrices $\mathbf{U}$ and $\mathbf{V}$ are orthonormal.

Different topic modeling techniques choose different strategies to conduct the matrix factorization. LSI [58] assumes that the K columns of matrix $\mathbf{U}$ as well as the K columns of matrix $\mathbf{V}$ are orthonormal. LSI amounts to minimizing the following objective function with the orthonormality constraints.

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{D} - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\|_F$$
$$\text{s.t. } \mathbf{U^T} \times \mathbf{U} = \mathbf{I}, \mathbf{V} \times \mathbf{V^T} = \mathbf{I}, \text{ and } \mathbf{\Sigma} \text{ is diagonal.}$$

Singular value decomposition (SVD) can be employed to solve the optimization problem, as shown in Figure 6.4.

NMF [111] is also a popular technique for topic modeling. NMF assumes that all the elements in term-document matrix $\mathbf{D}$ as well as the elements in term-topic matrix $\mathbf{U}$ and topic-document matrix $\mathbf{V}$ are nonnegative, as shown in Figure 6.5. Thus, the objective function of NMF is as follows.

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{D} - \mathbf{U}\mathbf{V}^T\|_F$$
$$\text{s.t.} \quad u_{ik} \geq 0, v_{jk} \geq 0.$$

The optimization can be solved by an iterative algorithm [111].

Another natural assumption in topic modeling is that the topics are sparse. RLSI [180, 179] exactly takes the assumption, as shown in Figure 6.5. Specifically, the objective function of RLSI is defined as

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{D} - \mathbf{U}\mathbf{V}^T\|_F + \lambda_1 \sum_{k=1}^{K} \|u_k\|_1 + \lambda_2 \sum_{n=1}^{N} \|v_n\|_2^2,$$

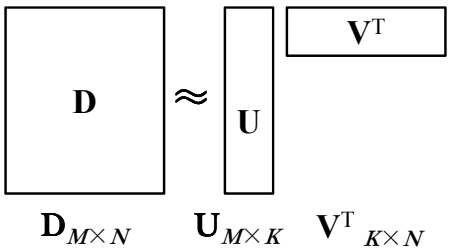

Figure 6.5: In non-negative matrix factorization, all elements in $\mathbf{D}$, $\mathbf{U}$, and $\mathbf{V}$ are nonnegative. In regularized latent semantic indexing, matrix $\mathbf{U}$ is assumed to be sparse.

where the ℓ_1 norm on $\mathbf{U}$ makes the topics sparse and the ℓ_2 norm on $\mathbf{V}$ makes the topic representations of documents smooth. The problem can also be solved by an iterative algorithm [180, 179]. One advantage of RLSI is that the optimization can be easily decomposed and performed in parallel, and thus is more scalable and efficient, when compared to LSI.

Non-probabilistic topic models have also been employed in a number of text processing tasks. For example, Xu et al. [195] propose conducting document clustering using NMF. Their method takes a topic discovered by NMF as a cluster, and assigns a document to the cluster (topic) for which the document has the largest projection value. Choi et al. [47] apply LSI to text segmentation. In their method they leverage LSI to estimate the inter-sentence similarity. Landauer et al. [107] propose a method of visualizing text information on the basis of LSI. They represent a document with topics obtained from LSI in a high dimensional dynamic viewer.

6.1.3 Probabilistic Interpretation

LSI, NMF, and RLSI can be interpreted within a probabilistic framework, as shown in Figure 6.6. In the framework, the columns of term-topic matrix u_k are assumed to be independent from each other and the columns of topic-document matrix v_n are regarded as latent variables. Each document d_n is assumed to be generated according to a Gaussian distribution conditioned on $\mathbf{U}$ and v_n, that

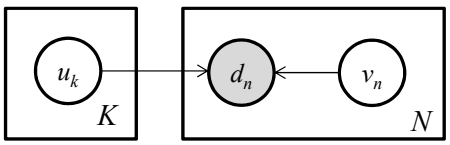

Figure 6.6: Graphical representation of non-probabilistic topic models.

Table 6.1: Prior/constraints in different non-probabilistic models.

method	prior/constraint on u_k	prior/constraints on v_n
LSI	orthonormality	orthonormality
NMF	$u_{mk} \geq 0$	$v_{kn} \geq 0$
RLSI	$P(u_k) \propto \exp(-\lambda_1 \|u_k\|_1)$	$P(v_n) \propto \exp(-\lambda_2 \|v_n\|_2^2)$

is, $P(d_n|\mathbf{U}, v_n) \propto \exp(-\|d_n - \mathbf{U}v_n\|_2^2)$. Furthermore, all pairs (d_n, v_n) are conditionally independent given $\mathbf{U}$.

Different techniques in fact use different priors or constraints on u_k's and v_n's. Table 6.1 lists the priors or constraints used in LSI, NMF, and RLSI, respectively. It can be shown that LSI, NMF, and RLSI can be obtained with maximum a posteriori (MAP) estimation in the framework.

6.2 Methods of Matching with Topic Model

Topic models can help deal with term mismatch in search. If the query contains the term "college" and the document contains the term "school", then there is a mismatch between the two terms, and the document may not be regarded relevant to the query. It is very likely that the two terms are included in the same topic, however, and the query and document can match in the topic space, as shown in Table 6.2. Previous work shows that the uses of topic models can indeed improve search relevance [58, 184, 200, 180, 179].

6.2.1 Linear Combination with Term Model

In practice, it is beneficial to combine topic matching scores with term matching scores to leverage the broadness of topic matching and

Table 6.2: Example topics (only top 5 words are shown for each topic).

topic 1	topic 2	topic 3	topic 4	topic 5	topic 6
OPEC	Africa	contra	school	Noriega	firefight
oil	South	Sandinista	student	Panama	ACR
cent	African	rebel	teacher	Panamanian	forest
barrel	Angola	Nicaragua	education	Delval	park
price	apartheid	Nicaraguan	college	canal	blaze

specificity of term matching. A simple and effective approach is to use a linear combination of term matching score and topic matching score, as proposed in [88]. The final relevance score $s(q, d)$ is calculated as follows.

$$s(q, d) = \alpha s_{topic}(q, d) + (1 - \alpha) s_{term}(q, d),$$

where $\alpha \in [0, 1]$ is a coefficient, $s_{term}(q, d)$ is the term matching score, and $s_{topic}(q, d)$ is the topic matching score. Term matching scores can be calculated with term-based models such as VSM and BM25. There are several ways to calculate topic matching scores. We classify existing methods into two categories: topic matching and smoothing.

6.2.2 Topic Matching

The topic matching score $s_{topic}(q, d)$ can be calculated in the following way. Given a query and a document, we first represent them in the topic space with a topic model. For example, in RLSI, query q can be represented in the topic space as

$$v_q = \arg\min_v \|q - \mathbf{U}v\|_2^2 + \lambda_2 \|v\|_2^2,$$

where q is the representation of the query in the word space (e.g., tf-idf). The solution to the optimization is

$$v_q = \left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1} q.$$

Note that $\left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1}$ can be calculated in advance off-line and thus the online inference can be made very fast.

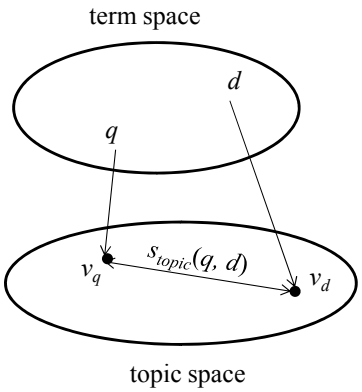

Figure 6.7: Topic matching.

Similarly, document d can be represented in the topic space as

$$v_d = \left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1} d.$$

The topic matching score between the query and the document in the topic space may be calculated as

$$s_{topic}(q, d) = \frac{\langle v_q, v_d\rangle}{\|v_q\|_2\|v_d\|_2}.$$

Figure 6.7 illustrates mapping a query and a document into the topic space for calculating the matching score.

In LDA, given query q, its topic representation is inferred as v_q, and given document d, its topic representation is inferred as v_d. Here both v_q and v_d are probabilistic distributions over topics. The topic matching score between the query and the document may be calculated as, for example, one minus the symmetric Kullback-Leibler divergence (abbreviated as KL divergence) between their topic distributions [88].

$$\begin{aligned} s_{topic}(q, d) &= 1 - \frac{1}{2}\left(\mathrm{KL}(v_q\|v_d) + \mathrm{KL}(v_d\|v_q)\right) \\ &= 1 - \frac{1}{2}\sum_{k=1}^{K}\left((v_q^k - v_d^k)\log\frac{v_q^k}{v_d^k}\right), \end{aligned}$$

where v_q^k and v_d^k are the k-th elements of v_q and v_d, respectively.

6.2.3 Smoothing

Probabilistic topic models can be utilized to smooth document language models (or query language models). A document language model can be used to represent the matching score between query and document

$$P(q|d) = \prod_{w \in q} P(w|d),$$

where $P(w|d)$ is the probability of word w given document d. Smoothing is necessary for LM4IR, because queries and documents are sparse, and thus many probabilities will be zero if maximum likelihood estimation (MLE) is employed.

Probability $P_{LM}(w|d)$ (the probability estimated with the language model) can be smoothed with a topic model [106, 59, 184, 200].

$$P(w|d) = \alpha P_{LM}(w|d) + (1 - \alpha) P_{TM}(w|d),$$

where $\alpha \in [0, 1]$ is a coefficient and $P_{TM}(w|d)$ is the probability estimated with the topic model. $P_{TM}(w|z)$ can be defined as

$$P_{TM}(w|d) = \sum_{z} P(w|z) P(z|d),$$

where z denotes a topic, $P(w|z)$ is the probability of word w in topic z, and $P(z|d)$ is the probability of topic z in document d. The final matching score between query q and document d becomes

$$\begin{aligned} P(q|d) &= \prod_{w \in q} P(w|d) \\ &= \prod_{w \in q} \left(\alpha P_{LM}(w|d) + (1 - \alpha) \left(\sum_{z=1}^{K} P(w|z) P(z|d) \right) \right). \end{aligned}$$

The technique can also be applied to a query model [200]. Given query q, the query language model is smoothed with topic model

$$P_{TM}(w|q) = \sum_{z} P(w|z) P(z|q),$$

where $P(w|z)$ is the probability of word w in topic z and $P(z|q)$ is the probability of topic z in query q. The use of $P_{TM}(w|q)$ can be interpreted as a type of query expansion.

Table 6.3: Performances of topic models in search.

	MAP	NDCG@1	NDCG@3	NDCG@5	NDCG@10
BM25	0.3918	0.4400	0.4268	0.4298	0.4257
BM25+LSI	0.3952	0.4720	0.4410	0.4360	0.4365
BM25+NMF	0.3985*	0.4600	0.4445*	0.4408*	0.4347*
BM25+PLSI	0.3928	0.4680	0.4383	0.4351	0.4291
BM25+LDA	0.3952	0.4760*	0.4478*	0.4332	0.4292
BM25+RLSI	**0.3998***	**0.4800***	**0.4461***	**0.4498***	**0.4420***

6.3 Experimental Results

We introduce some experimental results on using topic models for improving search relevance, reported in [181]. The experiment is conducted on TREC AP data, consisting of Associated Press articles. TREC topics 51∼300 are used as queries and relevance judgments are given at two levels: relevant or irrelevant. The parameter K (number of topics) is tuned on a validation set and the best performing parameter is selected for each model. Mean average precision (MAP) and normalized discounted cumulative gain (NDCG) at the positions of 1, 3, 5, and 10 are utilized as evaluation measures. Table 6.3 shows the performances achieved by combining BM25 and different topic models. Asterisks indicate significant improvements on the baseline of BM25 (t-test, $p < 0.05$). The results show that all topic models can improve the baseline. Among the topic models, RLSI, NMF, and LDA perform slightly better than the others.

7

Matching with Latent Space Model

In this section, we first introduce a general framework for matching between queries and documents in search. In the framework, queries in the query space and documents in the document space are mapped into a latent space. The matching function between query and document is then defined as the inner product between the images of query and document in the latent space. The framework is quite general and it contains traditional IR models as special cases. When the latent space has lower dimensionality than the original query space and document space, it will become capable to cope with query document mismatch. We next introduce five supervised learning methods for learning latent space models, in which the training data is derived from click-through data. The methods include partial least square (PLS), regularized mapping to latent space (RMLS), supervised semantic indexing (SSI), bilingual topic model (BLTM), and deep structured semantic model (DSSM). We also give some experimental results reported in previous works.

The major difference between the methods in Section 6 and the methods in this section is that the former are unsupervised learning algorithms and the latter are supervised learning algorithms.

Furthermore, the dimensions of the latent space in Section 6 usually represent topics, while the dimensions of the latent space in this section do not necessarily represent topics.

7.1 General Framework of Matching

We consider a general framework for matching queries and documents in search.

7.1.1 Latent Space

Suppose that there are two heterogenous spaces, query space and document space. The query space contains queries as objects and the document space contains documents as objects.[1] The dimensions of the query space and document space usually correspond to terms, more generally, n-grams of terms, or even other types of features [187]. Similarity measures are defined in the query space and document space, respectively. Without loss of generality, we assume that the measures are inner product. Suppose that there is a new space referred to as latent space. A similarity measure is also defined in the latent space, for example, inner product. We map the queries in the query space and documents in the document space into the latent space. We then utilize the similarity function in the new space to represent the matching degrees between queries and documents. In other words, matching between query and document is conducted in the latent space after mapping. Figure 7.1 illustrates the relations.

The latent space is specified by the two mapping functions. We can consider different types of mapping functions such as linear and non-linear mapping functions, as well as dimension-reduced and dimension-preserving mapping functions. The uses of different types of mapping functions lead to different types of matching models.

Next, we give a more formal definition of the framework. Here, $\mathcal{Q}$ denotes the query space and $\mathcal{D}$ denotes the document space.

[1] Query space and document space can be defined as the same space. It is more natural, however, to consider them as two different spaces, because queries and documents differ in nature.

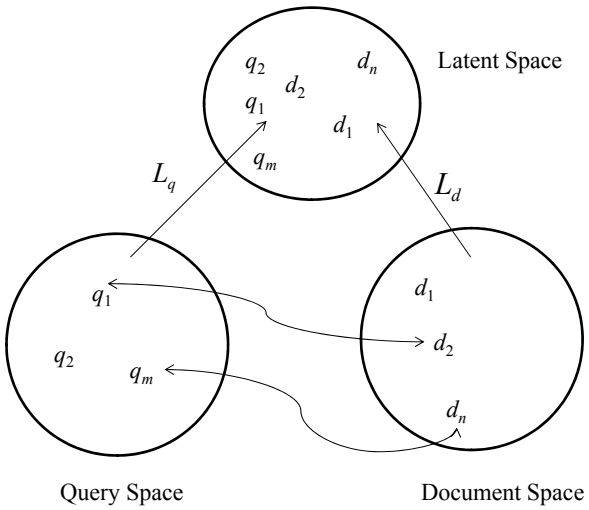

Figure 7.1: Matching in latent space.

Furthermore, q and d are query and document in $\mathcal{Q}$ and $\mathcal{D}$, respectively. $\mathcal{H}$ denotes the latent space. The mapping function from the query space to the latent space is represented as $\phi : \mathcal{Q} \mapsto \mathcal{H}$, where $\phi(q)$ stands for the mapped vector of q in $\mathcal{H}$. Similarly, the mapping function from the document space to the latent space is represented as $\phi' : \mathcal{D} \mapsto \mathcal{H}$, where $\phi'(d)$ stands for the mapped vector of d in $\mathcal{H}$. The matching function between q and d is defined as the inner product between $\phi(q)$ and $\phi'(d)$

$$k(q, d) = \langle \phi(q), \phi'(d) \rangle.$$

We refer to the matching function as latent space model in this survey.

7.1.2 Relation with Traditional IR Models

Traditionally, query document matching is represented in the IR models such as VSM [155], BM25 [153], and LM4IR [204]. VSM is a non-probabilistic model and BM25 and LM4IR are probabilistic models. All the models can be interpreted within the above matching framework.

If the latent space has the same dimensionality as the query and document spaces, and the two mappings are represented as diagonal matrices, then we obtain the VSM, BM25, and LM4IR models.

The VSM model is the simplest matching model within the framework. When the query space and document space are identical term spaces, and the two mapping functions are represented as two identity matrices, we obtain the VSM model

$$f_{\texttt{VSM}}(q,d) = \langle \phi_{\texttt{VSM}}(q), \phi'_{\texttt{VSM}}(d) \rangle = \langle q, d \rangle.$$

The BM25 model between query q and document d can be interpreted as the following matching function [193].

$$f_{\texttt{BM25}}(q,d) = \langle \phi_{\texttt{BM25}}(q), \phi'_{\texttt{BM25}}(d) \rangle,$$

where $\phi_{\texttt{BM25}}(q)$ and $\phi'_{\texttt{BM25}}(d)$ are two vectors in the latent space mapped from q and d. Each dimension of the vector corresponds to term x.

$$\phi_{\texttt{BM25}}(q)_x = \frac{(k_3+1)\cdot f(x,q)}{k_3 + f(x,q)}$$

and

$$\phi'_{\texttt{BM25}}(d)_x = \texttt{IDF}(x) \cdot \frac{(k_1+1)\cdot f(x,d)}{k_1\left(1-b+b\frac{f(d)}{avgf}\right) + f(x,d)},$$

where k_1, k_3, and b are parameters, $f(x,q)$ and $f(x,d)$ are the frequencies of term x in q and d, respectively, $f(d)$ is the total number of terms in document d, and $avgf$ is the average number of terms $f_t(d)$ per document. $\texttt{IDF}(x) = \log \frac{df - df(x) + 0.5}{df(x) + 0.5}$ is the inverse document frequency of term x, where $df(x)$ is the number of documents in which term x occurs and df is the total number of documents. BM25 thus is

$$f_{\texttt{BM25}}(q,d) = \sum_x \texttt{IDF}(x) \times \frac{(k_3+1)\times f(x,q)}{k_3 + f(x,q)} \times \frac{(k_1+1)\times f(x,d)}{k_1\left(1-b+b\frac{f(d)}{avgf}\right) + f(x,d)}.$$

LM4IR can also be interpreted in a similar way [193]. The traditional IR models are models representable by diagonal matrices, which means that the mappings preserve the dimensions of the original query and document spaces. As a result, terms can only match with themselves and cannot match with other terms in the same sense or topic. That is reason that the models suffer from term mismatch.

On the other hand, when the latent space has lower dimensionality than the original query space and document space and each dimension of the latent space represents a topic, the latent space model would have the capability of dealing with term mismatch. The question is how to learn the model (i.e., the two mapping functions).

7.2 Latent Space Models

We introduce five supervised learning methods for constructing latent space models: partial least square (PLS), regularized mapping to latent space (RMLS), supervised semantic indexing (SSI), bilingual topic model (BLTM) and deep structured semantic model (DSSM). The first four are linear models and the fifth is a non-linear model.

To learn a latent space model, we need training data indicating the matching relations between queries and documents across the two spaces. That is to say, this is a supervised learning problem. Fortunately, click-through data is accumulated in web search, and it can be naturally used as training data of the learning task, because it exactly gives the matching information. The training data is represented as $(q_1, d_1, c_1), (q_2, d_2, c_2), \cdots, (q_N, d_N, c_N)$. Each instance is a triple representing query, document, and click-number (or logarithm of click-number).[2].

With sufficient amount of training data and sophisticated machine learning techniques, we can really learn latent space models that are able to more accurately represent the matching relations between queries and documents and to deal with the mismatch problem. The dimensions of the latent space correspond to latent topics, if the dimensions of the query and document spaces correspond to terms. Intuitively, queries and documents strongly associated with a topic should have higher weights on the topic, and therefore they tend to match better in the latent space. The latent models are learned not only based on the information from the original query and document

[2]Click data is usually noisy. We follow the common practice and assume that we simply use the click numbers as training data (e.g., [71, 187, 90]) Certainly, how to improve the quality of the training data is an important research topic.

spaces, but also based on the information from the click-through data, and thus they can better represent the relevance relations in search.

The latent space models thus are learnable, more general, and more robust models, when compared with the traditional IR models.

7.2.1 Partial Least Square

PLS stands for partial least square, which is a technique originally proposed for regression in statistics. Another related technique is CCA (canonical correspondence analysis). For an introduction to PLS, see [154] and introduction to CCA, see [84].

PLS can be directly employed in learning of latent space model as shown in [187]. In PLS, the learning task is formalized as learning of two linear projection functions, i.e., two mapping functions which can be represented as orthonormal matrices. The matching function becomes

$$f(q, d) = \langle L_q \cdot q, L_d \cdot d \rangle. \tag{7.1}$$

The learning problem becomes an optimization problem as follows.

$$\arg\max_{L_q, L_d} = \sum_{(q_i, d_i)} c_i f(q_i, d_i),$$

$$L_q L_q^T = I, \quad L_d L_d^T = I$$

where (q_i, d_i) is a pair of query and document, c_i is the click number of the pair, L_q and L_d are projection functions (orthonormal matrices).

This is a non-convex optimization problem. It can be shown, however, that the global optimum exists and one can employ singular value decomposition (SVD) to solve the problem [187, 188].

PLS can also be viewed as learning of matching model from a click-through bipartite graph with feature vectors attached on the nodes or from a click-through matrix with feature vectors associated with the rows and columns.

The task can also be degenerated into learning of matching model solely from a click-through graph or a click-through matrix. The matching function can then be learned by directly applying SVD on the matrix (this is similar to latent semantic indexing (LSI)).

7.2.2 Regularized Mapping to Latent Space

It is hard to learn PLS when the data size is large, because we need to solve SVD, which is of high time complexity. Wu et al. [188] propose a new method called regularized mapping to latent space (RMLS) to address the issue. Specifically, they replace the orthonormality assumption in the formulation of PLS with a sparsity assumption to make RMLS. In this way, one can conduct the optimization in parallel and thus make the learning algorithm scalable. In [202, 71], Yih et al. and Gao et al. propose a similar model called discriminative projection model (DPM), where the learning is conducted based on Siamese neural network.

The optimization problem of RMLS is defined as follows.

$$\arg\max_{L_q, L_d} = \sum_{(q_i, d_i)} c_i f(q_i, d_i), \tag{7.2}$$

$$|l_q| \leq \theta_q, \quad |l_d| \leq \theta_d, \quad \|l_q\| \leq \tau_q, \quad \|l_d\| \leq \tau_d \tag{7.3}$$

where (q_i, d_i) is a pair of query and document, c_i is the click number of the pair, L_q and L_d are linear mapping functions, l_q and l_d are row vectors of L_q and L_d, and θ_q, θ_d, τ_q and τ_d are thresholds. $|\cdot|$ and $\|\cdot\|$ denote ℓ_1 and ℓ_2 norms. Note that the regularizations are defined on the row vectors, not column vectors. The use of ℓ_2 norm is to avoid a trivial solution.

There is no guarantee that there exists a global optimal solution for RMLS. One can employ a greedy algorithm to conduct the optimization. That is to first fix L_q and optimize L_d, and then to fix L_d and optimize L_q, and to repeat the process until convergence. From the formalization, one can verify that the optimization can be easily decomposed and performed row by row and column by column of the matrices. That is why the algorithm can be easily parallelized.

7.2.3 Supervised Semantic Indexing

Supervised semantic indexing (SSI), developed by Bai et al. [7, 8], takes the same view on the problem. First, the latent space model in (7.1) is rewritten as

$$f(q, d) = (L_q q)^T (L_d d) = q^T (L_q^T L_d) d.$$

Let $W = L_q^T L_d$, then we have

$$f(q, d) = q^T W d.$$

SSI tries to factorize W as follows

$$W = U^T V + I,$$

where I denotes the identity matrix. If $W = I$, then the model degenerates to VSM. If $W = U^T V$, then the model is equivalent to the model of PLS and RMLS.

SSI takes preference pairs derived from click-through log as training data, and utilizes hinge loss function as loss and gradient descent as optimization algorithm. No regularization is imposed on the loss function like that in RMLS.

7.2.4 Bilingual Topic Model

Bilingual topic model (BLTM) is a probabilistic model proposed by Gao et al. [71], which gives rise to the query-document pairs in click-through data, where documents are represented by their titles. The basic assumption of BLTM is that each query document pair is generated from the same distribution of topics. Figure 7.2 gives the graphical representation of the model.

The generative process of the model is as follows.

1. Each topic z is represented by a distribution of query words ϕ_q and a distribution of document words ϕ_d. The distributions are selected with Dirichlet priors with parameters β_q and β_d. There are K topics.

2. For each query-document pair q and d, a distribution of topics θ is selected with Dirichlet prior with parameter α. There are N query-document pairs.

3. In generation of each query, topic z is first selected according to distribution θ, and then query word w_q is selected according to distribution ϕ_q of topic z. There are $|q|$ query words.

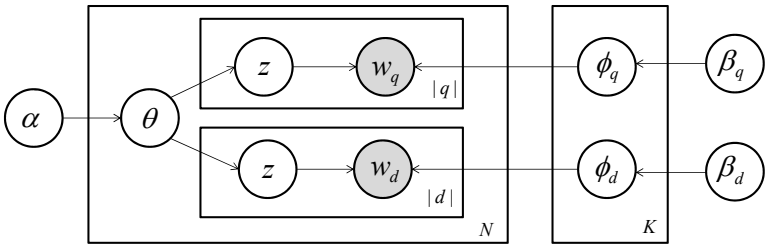

Figure 7.2: Graphical representation of bilingual topic model.

4. In generation of each document, topic z is first selected according to distribution θ, and then document word w_d is selected according to distribution ϕ_d of topic z. There are $|d|$ document words.

The expectation maximization (EM) algorithm can be employed to estimate the parameters of the model. The topic distributions of query words and document words form the 'latent space'.

The major difference between BLTM and the topic models of probabilistic latent semantic indexing (PLSI) and latent Dirichlet allocation (LDA) is that the former is a model of generating pairs of documents, while the latter are models of generating single documents. In fact, BLTM is a natural extension of LDA.

7.2.5 Deep Structured Semantic Model

Semantic matching between query and document may be better represented by non-linear models than linear models, because the semantic relations between query and document should be quite complex. Recently, Huang et al. propose performing semantic matching with deep learning techniques, and a model referred to as deep structured semantic model (DSSM) has been developed [90].

DSSM represents query q and its associated documents d's as vectors of terms and takes the vectors as input. It maps the input vectors into output vectors of lower dimensions through a multi-layer neural network, where the output vectors represent vectors of hidden topics. It then takes the cosine similarities between the output vector

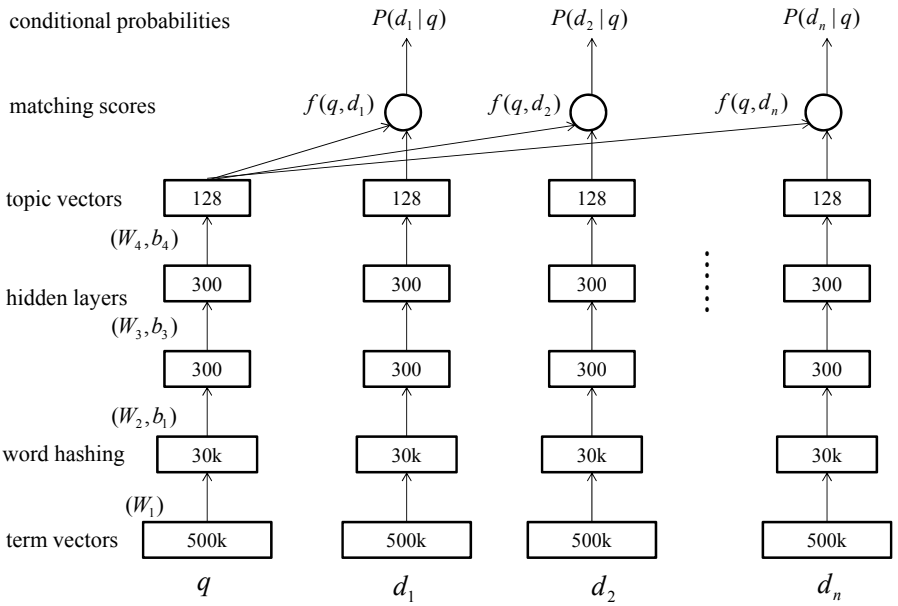

Figure 7.3: Deep Structured Semantic Model.

of query and output vectors of documents as matching scores between query and documents $f(q, d)$. Conditional probabilities $P(d|q)$ are calculated based on the matching scores $f(q, d)$ and click-through data is taken as observations of the conditional probabilities. Figure 7.3 shows the architecture of the deep network model.

The size of term vectors is very large in web search, DSSM employs a technique called word hashing to map term vectors to letter n-gram vectors to solve the problem, which corresponds to a fixed linear transformation. For example, word "good" is mapped into letter trigrams: (#go, goo, ood, od#), where # denotes starting and ending marks. In this way, the dimension of the vectors can be reduced from 500k to 30k, because the number of letter n-grams in English is limited.

Suppose that x denotes an input vector of terms, either representing a query or a document, and y denotes an output vector, representing a number of topics. Suppose that $i = 1, \cdots, k$ denote the intermediate layers, W_i denotes the i-th weight matrix and b_i denotes the i-th bias.

Then, the deep neural network model is defined as

$$\begin{aligned} l_1 &= W_1 x \\ l_i &= h(W_i l_{i-1} + b_i), i = 2, \cdots, k-1 \\ y &= h(W_n l_{k-1} + b_k), \end{aligned}$$

where h is the activation function defined as $tanh$ function,

$$h(x) = \frac{1 - \exp\{-2x\}}{1 + \exp\{-2x\}}.$$

The matching score between query q and document d is defined as

$$f(q, d) = \cos(y_q, y_d),$$

where y_q and y_d are output vectors of query q and document d. The conditional probability of document d given query q is defined as

$$P(d|q) = \frac{\exp\{\lambda f(q, d)\}}{\sum_{d \in D} \exp\{\lambda f(q, d)\}},$$

where D denotes a set of documents associated and λ is a parameter.

In learning, the model parameters are estimated by maximum likelihood estimation (MLE), with queries and their associated clicked documents as training data. Specifically the negative log likelihood function is minimized

$$L = -\log \prod_{(q,d) \in (Q,D)} P(d|q),$$

where (Q, D) denotes a set of queries and their associated documents (clicked documents and some randomly selected unclicked documents). A gradient based optimization technique is employed to conduct the minimization.

7.3 Experimental Results

We first introduce the experimental results in [188] in which the performances of different linear latent models are compared, including PLS, RMLS, BLTM, SSI, and SVDFeature. Here, SVDFeature [45] is

Table 7.1: Performances of latent space models in search.

	NDCG@1	NDCG@3	NDCG@5
BM25 (baseline)	0.637	0.690	0.690
SSI	0.538	0.621	0.629
SVDFeature	0.663	0.720	0.727
BLTM	0.657	0.702	0.701
PLS	0.676	0.728	**0.736**
RMLS	**0.686**	**0.732**	0.729

Table 7.2: Performances of latent space models in search.

	NDCG@1	NDCG@3	NDCG@10
BM25 (baseline)	0.308	0.373	0.455
WTM	0.332	0.400	0.478
LSI	0.298	0.372	0.455
PLSI	0.295	0.371	0.456
BLTM	0.337	0.403	0.480
DSSM (linear)	0.357	0.422	0.495
DSSM (non-linear)	**0.362**	**0.425**	**0.498**

similar to RMLS but for collaborative filtering and with a different loss function. There are 94,022 queries and 111,631 documents. Click through data associated with the queries and documents at a search engine is also used. Relevance judgments are made at five levels, and normalized discounted cumulative gain (NDCG) scores at the positions of 1, 3, and 5 are utilized as evaluation measures.

From Table 7.1, one can see that most of the latent models, trained from click-through data, work better than the baseline of BM25. Among them, RMLS and PLS perform the best. RMLS and PLS are comparable. Both of them significantly outperform the other methods (t-test, $p < 0.05$).

We next introduce the experimental results reported in [90], on comparison between different models including translation, topic, and latent space models. There are 16,510 queries and each query is on average associated with 15 web pages. Only the title fields of documents

are used for training and evaluation. Click-through data is randomly sampled for the high frequency query-document pairs for training, and evaluation is conducted on the low frequency query-document pairs in terms of NDCG.

The results in Table 7.2 show that DSSM works better than its linear version, as well as BLTM. The latent space models outperform the topic models of LSI and PLSI, as well as the translation model of WTM. [3]

[3]Significance testing results are not reported.

8

Learning to Match

So far we have seen many machine learning methods for matching queries and documents in search. They can be generalized as a more general machine learning problem, which we call 'learning to match'. Learning to match is applicable in many applications such as collaborative filtering (recommender systems), paraphrasing & textual entailment, as well as search. This section first gives a formal definition of learning to match. It then introduces methods of learning to match developed for collaborative filtering and paraphrasing & textual entailment. Finally, it makes discussions on potential applications of the techniques to search.

8.1 General Formulation

We give a formal definition of the learning to match problem as follows.

Suppose that there are two spaces X and Y. Objects x and y belong to the two spaces X and Y, respectively. A class of functions $F = \{f(x, y)\}$ is defined, referred to as class of matching functions. Training data $\{(x_1, y_1, r_1), \cdots, (x_N, y_N, r_N)\}$ is given, where each instance consists of objects x and y as well as their matching degree

r. The data is assumed to be generated according to the distributions $x \sim P(X), y \sim P(Y|X), r \sim P(R|X,Y)$. The goal of the learning task is to select a matching function $f(x,y)$ from the class on the basis of the training data. The matching function can be used in classification, regression, or ranking. The learning task, then, becomes the following optimization problem.

$$\arg\min_{f \in F} \sum_{i=1}^{N} L(r_i, f(x_i, y_i)) + \Omega(f),$$

where $L(r, f(x,y))$ denotes a loss function and Ω denotes a regularization.

What makes the task of learning to match unique is that it is to learn a two-input function $f(x,y)$, in contrast to learning of a one-input function. There usually exist relations between the two inputs x and y, and we can and should leverage the relations to enhance the accuracy of learning.

Many problems can be regarded as special applications of learning to match. These include collaborative filtering [101, 1, 2, 102, 45, 149, 9, 44], image annotation [83, 76, 159], paraphrasing and textual entailment [62, 54, 139, 203, 55, 85, 165, 35], question answering [37, 126, 21, 205, 178, 130], cross-language information retrieval [64, 141, 140], short text conversation [176, 130], similar document detection [32, 33, 42, 56, 199], online advertising [31, 30, 87], link prediction [124, 134], and drug design (matching between and receptor and ligand) [61].[1] A general model of matching is also proposed [75].

8.2 Methods of Collaborative Filtering

In recommendation, matching between users and products is performed. The so-called collaborative filtering approach is popular, which leverages the relations among users and the relations among products. This is a problem similar to query document matching in search.

[1]Similar relation between matching and ranking also exists in recommendation. For example, [9] proposes conducting matching first and then ranking in recommendation.

Latent factor models are state-of-the-art methods for collaborative filtering [1, 2]. They are basically models for performing matching between users and items (products) in latent spaces.

Koren proposes factorized neighborhood model (FNM) for collaborative filtering, which enhances the traditional neighborhood model by factorizing the user-user and item-item similarities in the model [101, 102]. Specifically, the unseen rating $\hat{r}_{ui}$ of item i by user u is defined as

$$\hat{r}_{ui} = b_0 + b_u + b_i + \frac{1}{|R(u)|^{\frac{1}{2}}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) q_i^T x_j + \frac{1}{|R(i)|^{\frac{1}{2}}} \sum_{v \in R(i)} (r_{vi} - b_{vi}) p_u^T y_v,$$

where $R(u)$ denotes the set of items rated by user u, $R(i)$ denotes the set of users giving rate to item i, b_0, b_u, b_i are biases, r_{uj} and r_{vi} denote the rates of item j by user u and item i by user v respectively, b_{uj} and b_{vi} are biases of item j by user u and item i by user v respectively, $q_i^T x_j$ and $p_u^T y_v$ are factorized similarities for items i-j and users u-v respectively. Intuitively, the fourth term represents the average of weighted ratings of the other items by the user with factorized item-item similarities as weights, and the fifth term represents the average of weighed ratings of the item by the other users with factorized user-user similarities as weights. Given training data, their method utilizes squared loss as loss function with ℓ_2 regularization, and employs gradient descent as learning algorithm to learn the biases and factors.

Chen et al. [45] have developed a feature-based matrix factorization method for collaborative filtering, called SVDFeature. Although simple, the method is quite powerful; the system based on it performed the best among the systems at the KDD Cup 2011 competition. In SVDFeature, the rating $r(u, i)$ of item i by user u is defined as follows,

$$\hat{r}_{ui} = \langle l_g, g \rangle + \langle l_u, u \rangle + \langle l_i, i \rangle + \langle L_u \cdot u, L_i \cdot i \rangle,$$

where u denotes the user feature vector, l_u denotes the weight vector of user, i denotes the item feature vector, l_i denotes the weight vector of item, g denotes a global feature vector, l_g denotes a global weight vector, L_u denotes a user factorization matrix, and L_i denotes an item

factorization matrix. In learning of the model, squared loss is utilized as loss function, ℓ_2 norm is utilized for regularization, and gradient descent is employed for optimization. The method is very similar to regularized mapping to latent space (RMLS) for search (cf., (7.2)-(7.3)), while the major difference lies in the loss functions.

Rendle proposes factorization machines (FM) [148, 149], a model for collaborative filtering, with certain relations to support vector machines (SVM) with polynomial kernel. Like polynomial SVM, FM models the interactions between all the variables in the input; but unlike polynomial SVM, FM uses factorized parameterization to deal with data sparseness. First, user u and item i are combined into one input vector x. Then the problem becomes to make classification, regression, or ranking prediction on vector x with the FM model $f(x)$.

$$f(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j, \tag{8.1}$$

where x is the input vector derived from a pair of user and item, n is the dimension of x, x_i and x_j are variables in the input, w_0 is a bias, w_i is the weight of variable x_i, and $\langle v_i, v_j \rangle$ is the weight of variables x_i and x_j, which can be factorized as

$$\langle v_i, v_j \rangle = \sum_{l=1}^{k} v_{i,l} \cdot v_{l,j},$$

where k denotes the number of factors. The model can be further extended from two-way interactions to d-way interactions, which corresponds to polynomial kernel $K(x, x') = (\langle x, x' \rangle + 1)^d$. Rendle shows that the FM model (8.1) can be computed efficiently with time complexity of $O(kn)$. FM generalizes polynomial SVM and can also mimic several existing models [148].

8.3 Methods of Paraphrasing & Textual Entailment

The issues of paraphrase detection and textual entailment detection can be formalized as matching between two strings, i.e., matching between two *structured objects*. In the former task, given two sentences one

wants to decide whether they are paraphrases of each other. In the latter task, given two sentences, one attempts to judge whether the first sentence implies the second sentence (e.g., "John goes to school every day" entails "John is a student"). We introduce three learning methods for matching of strings here.

Bu et al. [35, 36] introduce a class of string kernels for paraphrasing and textual entailment, which they call string rewriting kernel. With the use of the kernel function, existing kernel methods such as SVM can be directly employed for the task of string matching. String rewriting kernel measures the similarity between rewriting of string s_1 to t_1 and rewriting of string s_2 and t_2 with respect to all possible rewriting rules,

$$K((s_1, t_1), (s_2, t_2)) = \langle \Phi(s_1, t_1), \Phi(s_2, t_2) \rangle,$$

where s_1, t_1, s_2, and t_2 are strings, and $\Phi(s, t)$ denotes a feature vector representing scores for rewriting of s to t. Each element of feature vector $\phi_r(s, t)$ represents the score for rewriting of string s to string t with a rewriting rule r,

$$\Phi(s, t) = (\phi_r(s, t))_{r \in R}, \quad \phi_r(s, t) = n\lambda^i, \quad \lambda \in (0, 1],$$

where R denotes the set of rewriting rules, n the number of times rule r can be applied, i the number of wildcards in rule r, and λ a parameter. Figure 8.1 shows three examples of rewritings and one example of rewriting rule. The rewriting rule can be applied to the two rewritings on the top, but not the rewriting on the bottom. In principle, the string rewriting kernel measures how similar two rewritings are by applying all the possible rewriting rules (the number can be infinite).

Obviously, computation of the general kernel function is intractable. Bu et al. further propose a specific class of string rewriting kernels, referred to as kb-SRK, which can be computed efficiently. In kb-SRK, the rewriting rules are restricted to rewriting of k-grams and the wildcard alignments on the two strings are bijective (on the characters). They show that by using kb-SRK one can achieve the best performances on the benchmark datasets of paraphrasing and textual entailment.

Moschitti and Zanzotto [138, 139, 203] define a class of syntactic tree kernels, referred to as tree rewriting kernel in this survey, for paraphrasing and textual entailment, in contrast with conventional

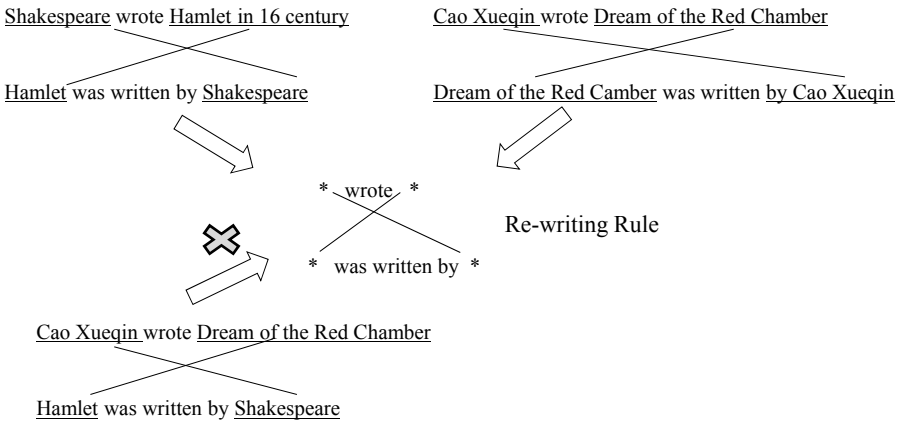

Figure 8.1: Examples of string rewriting and string rewriting rules.

tree kernels for tagging and parsing [49]. One can employ kernel methods such as SVM as well as the tree re-writing kernel to conduct the paraphrasing and textual entailment tasks. Given the syntactic trees of two pairs of sentences (T_1^α, T_2^α) and (T_1^β, T_2^β), a tree rewriting kernel measures the similarity between two pairs of sentences as the similarity between the rewritings of the syntactic trees of the two pairs, specifically, the rewriting from tree T_1^α to tree T_2^α and the rewriting from tree T_1^β to tree T_2^β. Figure 8.2 shows two rewritings of syntactic trees in textual entailment. First, placeholders are identified on the syntactic trees of each sentence pair to indicate the relations between the constituents in the rewriting; identical or synonymous constitutes (e.g., NP and VP) can be placeholders. For example, 1, 2, and 3 are placeholders on (T_1^α, T_2^α), and a, b, and c are placeholders on (T_1^β, T_2^β). Next, correspondences between the two rewritings are identified. For example, one correspondence is as follows. 1, 2, and 3 correspond to a, b, and c respectively. Then, the similarity between the two rewrittings is defined based on the similarity between trees T_1^α and T_1^β and the similarity between trees T_2^α and T_2^β with respect to all possible correspondences of placeholders.

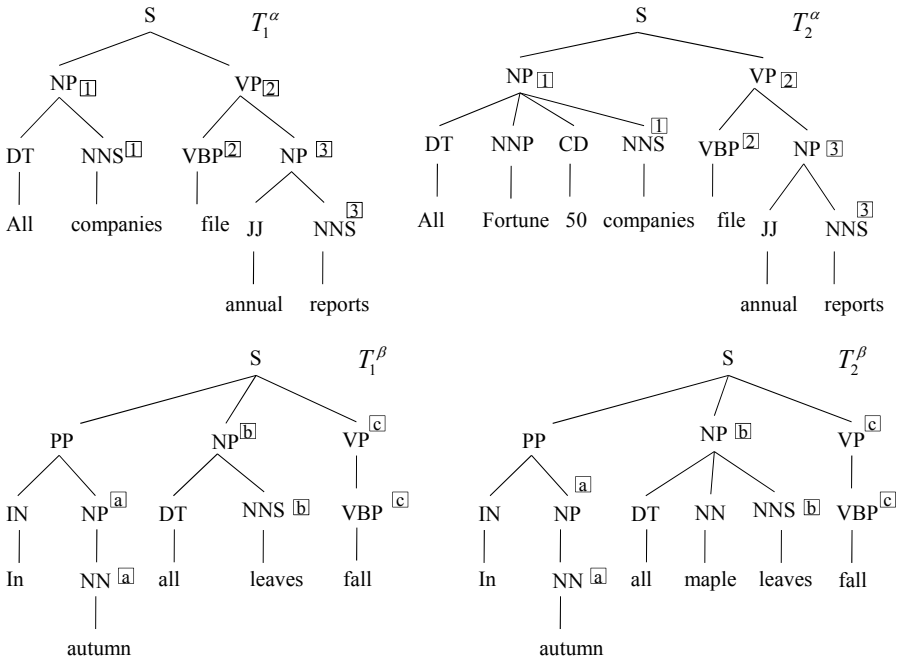

Figure 8.2: Examples of tree rewriting.

More formally, the kernel function is defined as

$$K_\Lambda\left((T_1^\alpha, T_2^\alpha), (T_1^\beta, T_2^\beta)\right) = \Lambda_{r\in R}\left(K_T\left(t(T_1^\alpha, r), t(T_1^\beta, r)\right) + K_T\left(t(T_2^\alpha, r), t(T_2^\beta, r)\right)\right),$$

where $K_T(\cdot,\cdot)$ denotes a tree kernel on two trees, $t(T, r)$ denotes a transformation of tree T with correspondence r on the placeholders of the tree, R denotes the set of all possible correspondences, and Λ stands for a function over R, e.g., sum or max. One possible correspondence is $r = \{(b, 1), (c, 2)\}$ in the figure. With this correspondence, the transformed trees $t(T_1^\alpha, r)$ and $t(T_1^\beta, r)$ can match and so can the transformed trees $t(T_2^\alpha, r)$ and $t(T_2^\beta, r)$.

Moschitti and Zanzotto propose an algorithm to efficiently compute a tree rewriting kernel. Their experimental results show that their

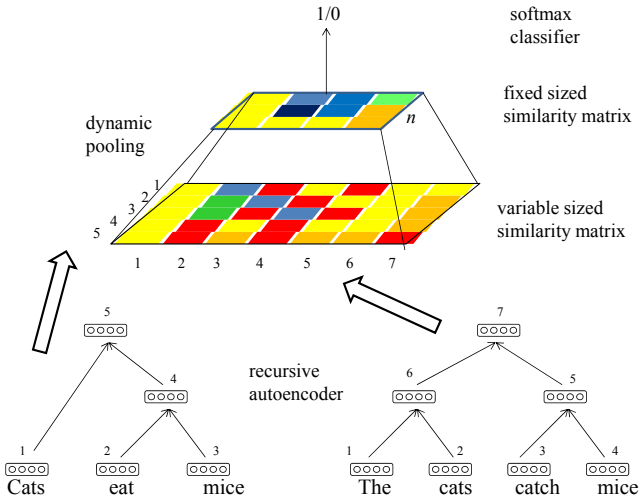

Figure 8.3: Paraphrase detection using unfolding recursive autoencoder and dynamic pooling.

method can obtain state-of-the-art accuracies in textual entailment and question answering.

Socher et al. [165] introduce a method for paraphrase detection using deep learning techniques. The basic idea is to parse the given two sentences and employ a neural network which can detect paraphrase relation by measuring the word-wise and phrase-wise similarities between the two sentences. The neural network is created by using two techniques, namely unfolding recursive autoencoders (URAE) and dynamic pooling. Figure 8.3 illustrates the process of creating the model.

First, the two sentences are parsed and represented as syntactic trees. Then, each word in the sentences is assigned a vector computed in advance by using neural language modeling. In neural language modeling [19] we jointly learn an embedding of words into a low-dimensional vector space and utilize the vectors to predict how likely a word occurs given a context. The vector of a word thus represents the 'semantics' of the word with its context.

Next, the vectors of the phrases (internal nodes) of each of the syntactic trees are constructed by an unfolding recursive autoencoder. Specifically, given a syntactic tree, the vector of each internal node in the tree is recursively computed by using the vectors of its child node as in a neural network. To assess how well the learned vectors of parent nodes represent their subtrees, the vectors are utilized to reconstruct the vectors of their subtrees and the reconstructed leaf vectors are compared with the original input vectors, and reconstruction errors in terms of Euclidean distance are calculated. The objective of the learning process is to minimize the reconstruction errors over all training data, and the parameters of the model (autoencoder), including the vectors of phrases, are learned by the minimization (cf., the lower part of Figure 8.3).

Finally, the Euclidean distances between all word and phrase vectors of the two sentences are computed, yielding a similarity matrix between the two sentences, where the rows and columns correspond to the nodes (word vectors and phrase vectors) in the two trees respectively, placed in pre-defined orders. If similar words align well in the two sentences, then the elements close to the diagonal of the matrix will tend to be zero. Because different sentence pairs have different sizes of similarity matrices, they cannot be directly fed it into a neural network model and normalization of the matrices needs to be performed. This is exactly what dynamic pooling does. The pooling method partitions the rows and columns of the similarity matrix into roughly equal parts and creates a grid covering the entire matrix. It then takes the minimum value of each cell and forms a new matrix consisting of all the minimum values within the grid. The obtained fixed-sized matrix is then used in one layer of the neural network for classification, i.e., detection (cf., the upper part of Figure 8.3).

8.4 Potential Applications to Search

The methods described above can be potentially applied to search, although they are originally developed for collaborative filtering and paraphrasing & textual entailment.

Collaborative filtering and search are similar in the sense that both are based on matching between heterogenous objects, while in the former task the objects are users and items and in the latter task the objects are queries and documents. Therefore, in principle technologies developed for collaborative filtering can be applied to search, and vice versa. For example, the method of SVDFeature developed for collaborative filtering is very similar to the method of RMLS developed for search. The only difference lies in the loss functions utilized by the two methods. More research can be conducted to investigate the relations between methods for the two applications.

Queries in web search are usually short. For example, about 90% of the queries are of length not exceeding five and about 10% of the queries are of length between five and twelve, in one search log dataset [14]. Furthermore, about 7% of the long queries are natural language questions. We may enhance the relevance ranking of natural language questions by employing the techniques for paraphrasing & textual entailment. More specifically, we extract the titles and key sentences of documents and store them in index in advance. When a query is submitted, we identify whether the query is a natural language question using a parser. If it is, then we conduct matching between the question and the titles and key sentences of documents, and use the matching results as features of the ranking model, where the matching is carried out using paraphrasing & textual entailment techniques. That is to say, matching between query and document on structure aspect is performed (e.g., "how far is sun from earth" and "distance between sun and earth").

9

Conclusion and Open Problems

9.1 Summary of Survey

Query document mismatch is one of the biggest challenges in search. The major reason is that the searchers and authors of documents may use different expressions to represent the same meaning. Due to the richness and flexibility of language this phenomenon can regularly happen in search. On the other hand, the main mechanism of search is still based on term level matching, without genuine language understanding. This results in most of the dissatisfactions in search in which information exists in the system but users cannot find it.

One effective approach to dealing with the challenge is to conduct more query understanding and more document understanding, and conduct matching between richer query representation and richer document representation which can better characterize the meanings of query and document, referred to as semantic matching in this survey. Queries and documents can be represented on form, phrase, sense, topic, and structure aspects, and so can matching be performed.

In this survey, we have introduced machine learning approaches to dealing with query document mismatch, particularly recent advances along this direction. Methods for matching by query reformulation,

matching with term dependency model, matching with translation model, matching with topic model, and matching with latent space model have been explained. One can see that machine learning for matching between query and document in search has made and is making significant progress. The problem of mismatch is far from being solved, however. More and more research on the issue is definitely needed in the future.

Matching between heterogenous objects from two spaces can be found in many applications, for example, collaborative filtering and paraphrasing & textual entailment. Machine learning techniques for matching, referred to as learning to match, need also be extensively and deeply studied. Learning to match methods for collaborative filtering and paraphrasing & textual entailment have been described in this survey, which are potentially applicable to search as well.

9.2 Comparison between Approaches

We next make comparisons between the five approaches described in Sections 3-7, namely, matching by query reformulation, matching with term dependency model, matching with translation model, matching with topic model, and matching with latent space model.

We must note that previous work on the approaches was done on different datasets with different properties, and there was no direct comparison between the approaches using a single dataset. This appears to be an important research topic in the future. We have observed in our own studies, however, that the approaches described in this survey tend to be complementary and thus it is usually advantageous to combine the uses of all the approaches.

Table 9.1 lists up the major characteristics of the five approaches, denoted as Query, Dependency, Translation, Topic, and Latent. The approaches are compared in terms of type of model, type of training data, computation complexity of learning. In practice one can choose approaches that are suitable to their problem settings.

In Section 2, we have given a system view of semantic matching. We next explain how the five approaches can be integrated into the

Table 9.1: Characteristics of approaches.

	model	training data	complexity of learning
Query	query	search log	small
Dependency	query-document	relevance	small
Translation	query-document	click-through	small
Topic	document	document	high
Latent	query-document	click-through	high

system. In query understanding, spelling errors in the query can be corrected using the query reformulation techniques described in Section 3.2, similar queries can be found by using the similar query finding techniques described in Section 3.3, topics of the query can be identified using the topic modeling techniques described in Section 6.2. In document understanding, topics of the document can be identified by using the topic modeling techniques as well. In query document matching, the different fields of query (the enriched representation of query) and different fields of document (the enriched document) can be matched, where matching can be carried out by the term dependency model in Section 4, the translation model in Section 5, and the latent space model in Section 7.

Matching results based on the term dependency model, translation model, topic model, and latent space model can be used as features of the ranking model, and learning to rank techniques can be used to train the ranking model.

9.3 Other Approaches

Query reformulation has been intensively studied so far. Similarly, 'document reformulation' might also be considered (i.e., to create additional representation for the document to make it have better matching with relevant queries). Document reformulation includes key phrase extraction [144, 175], key word or phrase annotation [196, 72, 201], document expansion [164, 114, 169]. For example, Xue et al. [196] conduct key word or phrase annotation to documents for

search using a click-through bipartite graph. The assumption is that two documents should be similar and assigned to similar queries if they have many co-clicks. An iterative algorithm has been developed in which the document similarities and query similarities are jointly calculated through propagation of similarity scores on the graph. The document similarities are further utilized for annotating key words and phrases to webpages and enhancing search relevance.

Query classification [97, 113, 13, 161, 119] and document classification should be helpful for addressing query document mismatch. For example, Bennett et al. [20] presents a classification method for web search. Their method categorizes web pages into semantic classes in advance. In search, given a query, the method derives the distribution of classes of the query using the search result as well as click-through data, and then takes the matching degrees between the class distributions of query and the class distributions of web pages as features of the ranking model. Experimental results show that the method can enhance the performance of search relevance. The advantage of the classification approach is that it is based on supervised learning approach and the classes can be more accurately learned, while the disadvantage is that the classes are predefined and cannot be dynamically adapted to data.

Named entity recognition in query [78, 190], in parallel, and named entity recognition in document should be helpful as well. For instance, Lu et al. [129] propose a method of enhancing search relevance by leveraging named entity recognition in query. Their method first makes use of a number of entity recognizers to identify different types of named entities in the query. It then utilizes the categories of identified named entities and the categories of sections of document to derive a number of semantic matching features. The matching features are incorporated into the ranking model which is trained with learning to rank techniques. For the query "san francisco college", the categories assigned to the query phrases can be "san francisco[city-name] college[business]". The authors observe that for long queries containing several named entities significant gain can be obtained in relevance ranking by employing their method.

Another approach is to enrich query and document representations with human knowledge such as Wikipedia and conduct semantic matching with the representations [65]. The approach is gaining attention recently, because more and more knowledge bases become available.

9.4 Open Problems and Future Directions

There are many open questions with regard to machine learning for semantic matching in search. Here, we only list some of them.

- Topic drift can easily occur in learning-based semantic matching, because language is by nature synonymous and polysemous. How to deal with the challenge is the key question for the machine learning paradigm. More powerful techniques surely need to be developed.
- Scalability is another important issue which needs to be considered. For example, learning of topic model and latent space model needs a large scale computing environment. This is particularly critical in the era of big data.
- Missing information in training data is a common issue. For rare queries and documents, it is difficult to collect sufficient training data. This in fact belongs to the long tail challenge. One possible way to conquer the challenge would be to incorporate existing knowledge such as WordNet and Wikipedia into the matching model. There has been some progress on research of the problem (e.g., [65, 104]), but certainly more studies are necessary.
- More natural language processing techniques need to be employed to enhance the performance of query document matching in search. This is because long queries and natural language queries are comprised of a large proportion of dissatisfied queries. Linguistic knowledge such as that indicating "distance between X and Y" and "how far is X from Y" are equivalent representations is necessary to be utilized in query document matching.

- The Cranfield approach [48] is usually employed for evaluation of a search system, particularly evaluation of relevance models. When semantic matching is necessary, it often involves cases in which the queries are tail queries and relevance judgments are difficult for human assessors, who are not query owners. The question is whether it is possible to introduce new mechanisms for relevance evaluation, particularly that using users' click data, to enhance the research on matching.

Acknowledgements

We would like to express our sincere gratitude to our former and current colleagues, interns, and collaborators, Gu Xu, Wei Wu, Quan Wang, Jiafeng Guo, Zhengdong Lu, Fan Bu, Tianqi Chen, Jingfang Xu, Daxin Jiang, Yunhua Hu, Ziqi Wang, Chen Wang, Haocheng Wu, Xiaobing Xue, Shuanghong Yang, Hao Wang, Keping Bi, Chaoliang Zhong, Yu Tao, W. Bruce Croft, Michael Bendersky, Xueqi Cheng, Xiaoyan Zhu, Yong Yu, Enhong Chen, Ming Zhang, Nick Craswell, and Satoshi Oyama, with whom we worked together on the problem of semantic matching in search and in general learning to match. This survey benefits a lot from the joint work with them.

We thank the three anonymous reviewers, as well as Douglas W. Oard, Quan Wang, Xin Jiang, Stephen Robertson, Jagadeesh Gorla, Milad Shokouhi, who read the draft of this survey and made many valuable comments.

We would also like to thank Douglas W. Oard and Mark Sanderson. Without their support and guidance, this survey would not have been published.

This work is supported in part by China National 973 project 2014CB340301.

References

[1] Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *J. Mach. Learn. Res.*, 10:803–826, June 2009.

[2] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.

[3] Farooq Ahmad and Grzegorz Kondrak. Learning a spelling error model from search query logs. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 955–962, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[4] Jaime Arguello, Jonathan L. Elsas, Jamie Callan, and Jaime G. Carbonell. Document representation and query expansion models for blog recommendation. In *International Conference on Weblogs and Social Media*, pages 10–18. AAAI Press, 2008.

[5] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query clustering for boosting web page ranking. In Jesĺšs Favela, Ernestina Menasalvas, and Edgar Chĺćvez, editors, *Advances in Web Intelligence*, volume 3034 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin Heidelberg, 2004.

[6] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search*. Pearson Education Ltd., Harlow, England, 2nd edition, 2011.

[7] Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. Supervised semantic indexing. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 187–196, New York, NY, USA, 2009. ACM.

[8] Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. Learning to rank with (a lot of) word features. *Inf. Retr.*, 13(3):291–314, June 2010.

[9] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 143–152, New York, NY, USA, 2012. ACM.

[10] Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R. Carvalho. Exploring reductions for long web queries. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 571–578, New York, NY, USA, 2010. ACM.

[11] Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann. Broccoli: Semantic full-text search at your fingertips. *CoRR*, 2012.

[12] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 407–416, New York, NY, USA, 2000. ACM.

[13] Steven M. Beitzel, Eric C. Jensen, Ophir Frieder, David Grossman, David D. Lewis, Abdur Chowdhury, and Aleksandr Kolcz. Automatic web query classification using labeled and unlabeled training data. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 581–582, New York, NY, USA, 2005. ACM.

[14] Michael Bendersky and W. Bruce Croft. Analysis of long queries in a large scale search log. In *Proceedings of the 2009 Workshop on Web Search Click Data*, WSCD '09, pages 8–14, New York, NY, USA, 2009. ACM.

[15] Michael Bendersky and W. Bruce Croft. Modeling higher-order term dependencies in information retrieval using query hypergraphs. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 941–950, New York, NY, USA, 2012. ACM.

[16] Michael Bendersky, W. Bruce Croft, and David A. Smith. Joint annotation of search queries. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL '11, pages 102–111. The Association for Computer Linguistics, 2011.

[17] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 31–40, New York, NY, USA, 2010. ACM.

[18] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Effective query formulation with multiple information sources. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 443–452, New York, NY, USA, 2012. ACM.

[19] Yoshua Bengio, Holger Schwenk, Jean-Sĺębastien Senĺęcal, Frĺęderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In DawnE. Holmes and LakhmiC. Jain, editors, *Innovations in Machine Learning*, volume 194 of *Studies in Fuzziness and Soft Computing*, pages 137–186. Springer Berlin Heidelberg, 2006.

[20] Paul N. Bennett, Krysta Svore, and Susan T. Dumais. Classification-enhanced ranking. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 111–120, New York, NY, USA, 2010. ACM.

[21] Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 192–199, New York, NY, USA, 2000. ACM.

[22] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 222–229, New York, NY, USA, 1999. ACM.

[23] Shane Bergsma and Qin Iris Wang. Learning noun phrase query segmentation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 819–826, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[24] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[25] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[26] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[27] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: Model and applications. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 609–618, New York, NY, USA, 2008. ACM.

[28] Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 211–218, New York, NY, USA, 2002. ACM.

[29] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[30] Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich, Vanja Josifovski, Donald Metzler, Lance Riedel, and Jeffrey Yuan. Online expansion of rare queries for sponsored search. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 511–520, New York, NY, USA, 2009. ACM.

[31] Andrei Broder, Marcus Fontoura, Vanja Josifovski, and Lance Riedel. A semantic approach to contextual advertising. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 559–566, New York, NY, USA, 2007. ACM.

[32] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.

[33] Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, COM '00, pages 1–10, London, UK, UK, 2000. Springer-Verlag.

[34] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993.

[35] Fan Bu, Hang Li, and Xiaoyan Zhu. String re-writing kernel. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 449–458, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[36] Fan Bu, Hang Li, and Xiaoyan Zhu. An introduction to string re-writing kernel. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 2982–2986. AAAI Press, 2013.

[37] Robin D Burke, Kristian J Hammond, Vladimir Kulyukin, Steven L Lytinen, Noriko Tomuro, and Scott Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57, 1997.

[38] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 243–250, New York, NY, USA, 2008. ACM.

[39] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 875–883, New York, NY, USA, 2008. ACM.

[40] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50, January 2012.

[41] Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world-wide web. *Comput. Netw. ISDN Syst.*, 27(6):1065–1073, April 1995.

[42] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM.

[43] Qing Chen, Mu Li, and Ming Zhou. Improving query spelling correction using web search results. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 181–189. ACL, 2007.

[44] TianQi Chen, Hang Li, Qiang Yang 0001, and Yong Yu. General functional matrix factorization using gradient boosting. In *ICML '13: Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Proceedings*, pages 436–444, 2013.

[45] Tianqi Chen, Zhao Zheng, Qiuxia Lu, Weinan Zhang, and Yong Yu. Feature-based matrix factorization. *CoRR*, abs/1109.2271, 2011.

[46] David Chiang. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June 2007.

[47] Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. Latent semantic analysis for text segmentation. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 109–117, 2001.

[48] C.W. Cleverdon. *The Effect of Variations in Relevance Assessment in Comparative Experimental Tests of Index Languages*. Cranfield Library report. Cranfield Inst. of Technology, 1970.

[49] Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[50] Nick Craswell and Martin Szummer. Random walks on the click graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 239–246, New York, NY, USA, 2007. ACM.

[51] W. Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. Query representation and understanding workshop. *SIGIR Forum*, 44(2):48–53, January 2011.

[52] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 1st edition, 2009.

[53] Silviu Cucerzan and Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, EMNLP '04, pages 293–300. ACL, 2004.

[54] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190. Springer-Verlag, Berlin, Heidelberg, 2006.

[55] Dipanjan Das and Noah A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 468–476, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[56] Ali Dasdan, Paolo D'Alberto, Santanu Kolay, and Chris Drome. Automatic retrieval of similar content using search engine query interface. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 701–710, New York, NY, USA, 2009. ACM.

[57] Fabio De Bona, Stefan Riezler, Keith Hall, Massimiliano Ciaramita, Amaç Herdağdelen, and Maria Holmqvist. Learning dense models of query similarity from user click logs. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 474–482, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[58] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[59] Fernando Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 672–679, New York, NY, USA, 2005. ACM.

[60] Fernando Diaz and Donald Metzler. Improving the estimation of relevance models using large external corpora. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 154–161, New York, NY, USA, 2006. ACM.

[61] Hao Ding, Ichigaku Takigawa, Hiroshi Mamitsuka, and Shanfeng Zhu. Similarity-based machine learning methods for predicting drugĺctarget interactions: a brief review. *Briefings in Bioinformatics*, page bbt056, 2013.

[62] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[63] Huizhong Duan and Bo-June (Paul) Hsu. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 117–126, New York, NY, USA, 2011. ACM.

[64] Susan T Dumais, Todd A Letsche, Michael L Littman, and Thomas K Landauer. Automatic cross-language retrieval using latent semantic indexing. In *AAAI spring symposium on cross-language text and speech retrieval*, volume 15, page 21, 1997.

[65] Ofer Egozi, Shaul Markovitch, and Evgeniy Gabrilovich. Concept-based information retrieval using explicit semantic analysis. *ACM Trans. Inf. Syst.*, 29(2):8:1–8:34, April 2011.

[66] Edward A. Fox and Joeseph A. Shaw. Combination of multiple searches. In *The Second Text REtrieval Conference (TREC-2)*, volume 500-215 of *NIST Special Publication*, pages 243–252. NIST, 1994.

[67] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971, November 1987.

[68] Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1139–1148, New York, NY, USA, 2010. ACM.

[69] Jianfeng Gao and Jian-Yun Nie. Towards concept-based translation models using search logs for query expansion. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1:1–1:10, New York, NY, USA, 2012. ACM.

[70] Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. Dependence language model for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 170–177, New York, NY, USA, 2004. ACM.

[71] Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. Clickthrough-based latent semantic models for web search. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 675–684, New York, NY, USA, 2011. ACM.

[72] Jianfeng Gao, Wei Yuan, Xiao Li, Kefeng Deng, and Jian-Yun Nie. Smoothing clickthrough data for web search ranking. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 355–362, New York, NY, USA, 2009. ACM.

[73] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *Proceedings of The Semantic Web: Research and Applications, First European Semantic Web Symposium*, ESWS '04, pages 61–75. Springer, 2004.

[74] Andrew R. Golding and Dan Roth. A winnow-based approach to context-sensitive spelling correction. *Mach. Learn.*, 34(1-3):107–130, February 1999.

[75] Jagadeesh Gorla, Stephen Robertson, Jun Wang, and Tamas Jambor. A theory of information matching. *CoRR*, abs/1205.5569, 2012.

[76] David Grangier and Samy Bengio. A discriminative kernel-based approach to rank images from text queries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(8):1371–1384, August 2008.

[77] R. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 700–709, New York, NY, USA, 2003. ACM.

[78] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 267–274, New York, NY, USA, 2009. ACM.

[79] Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 379–386, New York, NY, USA, 2008. ACM.

[80] Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. Towards optimum query segmentation: In doubt without. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1015–1024, New York, NY, USA, 2012. ACM.

[81] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. Query segmentation revisited. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 97–106, New York, NY, USA, 2011. ACM.

[82] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 362–370, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[83] David R. Hardoon and John Shawe-taylor. Kcca for different level precision in content-based image retrieval. In *In Third International Workshop on Content-Based Multimedia Indexing, IRISA*, 2003.

[84] David R. Hardoon, Sandor R. Szedmak, and John R. Shawe-taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.*, 16(12):2639–2664, December 2004.

[85] Michael Heilman and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[86] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *In Proceedings of the International Conference on Articial Neural Networks*, pages 97–102, 1999.

[87] Dustin Hillard, Stefan Schroedl, Eren Manavoglu, Hema Raghavan, and Chirs Leggetter. Improving ad relevance in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 361–370, New York, NY, USA, 2010. ACM.

[88] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[89] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *J. Am. Soc. Inf. Sci. Technol.*, 54(7):638–649, May 2003.

[90] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, New York, NY, USA, 2013. ACM.

[91] Samuel Huston, J. Shane Culpepper, and W. Bruce Croft. Indexing word sequences for ranked retrieval. *ACM Trans. Inf. Syst.*, 32(1):3:1–3:26, January 2014.

[92] Aminul Islam and Diana Inkpen. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1241–1249, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[93] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[94] Daxin Jiang, Jian Pei, and Hang Li. Mining search and browse logs for web search: A survey. *ACM Trans. Intell. Syst. Technol.*, 4(4):57:1–57:37, October 2013.

[95] Rong Jin, Alex G. Hauptmann, and Cheng Xiang Zhai. Title language model for information retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 42–48, New York, NY, USA, 2002. ACM.

[96] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA, 2006. ACM.

[97] In-Ho Kang and GilChang Kim. Query type classification for web document retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 64–71, New York, NY, USA, 2003. ACM.

[98] Maryam Karimzadehgan and ChengXiang Zhai. Estimation of statistical translation models based on mutual information for ad hoc information retrieval. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 323–330, New York, NY, USA, 2010. ACM.

[99] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[100] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[101] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[102] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24, January 2010.

[103] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[104] Alexander Kotov and ChengXiang Zhai. Tapping into knowledge base for concept feedback: Leveraging conceptnet to improve search results for difficult queries. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 403–412, New York, NY, USA, 2012. ACM.

[105] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68, New York, NY, USA, 2009. ACM.

[106] Oren Kurland and Lillian Lee. Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 194–201, New York, NY, USA, 2004. ACM.

[107] T. K. Landauer, D. Laham, and M. Derr. From paragraph to graph: Latent semantic analysis for information visualization. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5214–5219, apr 2004.

[108] Hao Lang, Donald Metzler, Bin Wang, and Jin-Tao Li. Improved latent concept expansion using hierarchical markov random fields. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 249–258, New York, NY, USA, 2010. ACM.

[109] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, New York, NY, USA, 2001. ACM.

[110] Matthew Lease. An improved markov random field model for supporting verbose queries. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 476–483, New York, NY, USA, 2009. ACM.

[111] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

[112] Joon Ho Lee. Analyses of multiple evidence combination. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '97, pages 267–276, New York, NY, USA, 1997. ACM.

[113] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 391–400, New York, NY, USA, 2005. ACM.

[114] Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. Dictionary-based techniques for cross-language information retrieval. *Inf. Process. Manage.*, 41(3):523–547, May 2005.

[115] Hang Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011.

[116] Hang Li. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94-D(10):1854–1862, 2011.

[117] Hang Li, Gu Xu, W. Bruce Croft, Michael Bendersky, Ziqi Wang, and Evelyne Viegas. Qru-1: A public dataset for promoting query representation and understanding research. In *Proceedings of the Workshop on Web Search Click Data*, WSCD '12, 2012.

[118] Mu Li, Yang Zhang, Muhua Zhu, and Ming Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 1025–1032, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[119] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 339–346, New York, NY, USA, 2008. ACM.

[120] Yanen Li, Huizhong Duan, and ChengXiang Zhai. A generalized hidden markov model with discriminative training for query spelling correction. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 611–620, New York, NY, USA, 2012. ACM.

[121] Yanen Li, Bo-Jun Paul Hsu, ChengXiang Zhai, and Kuansan Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 285–294, New York, NY, USA, 2011. ACM.

[122] Yinghao Li, Wing Pong Robert Luk, Kei Shiu Edward Ho, and Fu Lai Korris Chung. Improving weak ad-hoc queries using wikipedia as external corpus. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 797–798, New York, NY, USA, 2007. ACM.

[123] Zhen Liao, Daxin Jiang, Enhong Chen, Jian Pei, Huanhuan Cao, and Hang Li. Mining concept sequences from large-scale search logs for context-aware query suggestion. *ACM Trans. Intell. Syst. Technol.*, 3(1):17:1–17:40, October 2011.

[124] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, May 2007.

[125] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Nat. Lang. Eng.*, 7(4):343–360, December 2001.

[126] Kenneth C Litkowski. Question-answering using semantic relation triples. In *In Proceedings of the 8th Text Retrieval Conference (TREC-8)*, pages 349–356, 1999.

[127] H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October 2004.

[128] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.

[129] Yumao Lu, Fuchun Peng, Gilad Mishne, Xing Wei, and Benoit Dumoulin. Improving web search relevance with semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 648–657, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[130] Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1367–1375. Curran Associates, Inc., 2013.

[131] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[132] K. Tamsin Maxwell and W. Bruce Croft. Compact query term selection using topically related text. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 583–592, New York, NY, USA, 2013. ACM.

[133] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 469–478, New York, NY, USA, 2008. ACM.

[134] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'11, pages 437–452, Berlin, Heidelberg, 2011. Springer-Verlag.

[135] Donald Metzler. *A Feature-Centric View of Information Retrieval*. Springer, 2012 edition, 2011.

[136] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 472–479, New York, NY, USA, 2005. ACM.

[137] Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 311–318, New York, NY, USA, 2007. ACM.

[138] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 318–329, Berlin, Heidelberg, 2006. Springer-Verlag.

[139] Alessandro Moschitti and Fabio Massimo Zanzotto. Fast and effective kernels for relational learning from texts. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 649–656, New York, NY, USA, 2007. ACM.

[140] Jian-Yun Nie. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125, 2010.

[141] Douglas W Oard and Anne R Diekema. Cross-language information retrieval. *Annual Review of Information Science (ARIST)*, 33, 1998.

[142] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[143] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[144] Deepa Paranjpe. Learning document aboutness from implicit user feedback and document structure. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 365–374, New York, NY, USA, 2009. ACM.

[145] Jae Hyun Park, W. Bruce Croft, and David A. Smith. A quasi-synchronous dependence model for information retrieval. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 17–26, New York, NY, USA, 2011. ACM.

[146] Fuchun Peng, Nawaaz Ahmed, Xin Li, and Yumao Lu. Context sensitive stemming for web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 639–646, New York, NY, USA, 2007. ACM.

[147] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 160–169, New York, NY, USA, 1993. ACM.

[148] Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, IEEE Computer Society, 2010.

[149] Steffen Rendle. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[150] Stefan Riezler and Yi Liu. Query rewriting using monolingual statistical machine translation. *Comput. Linguist.*, 36(3):569–582, September 2010.

[151] Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532, May 1998.

[152] S. E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[153] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995.

[154] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *Proceedings of the 2005 international conference on Subspace, Latent Structure and Feature Selection*, SLSFS'05, pages 34–51, Berlin, Heidelberg, 2006. Springer-Verlag.

[155] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[156] Tefko Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, 26(6):321–343, 1975.

[157] Tefko Saracevic. Relevance: A review of the literature and a framework for thinking on the notion in information science. part iii: Behavior and effects of relevance. *J. Am. Soc. Inf. Sci. Technol.*, 58(13):2126–2144, November 2007.

[158] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[159] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S. Davis. Human detection using partial least squares analysis. In *IEEE 12th International Conference on Computer Vision*, pages 24–31. IEEE, 2009.

[160] Daniel Sheldon, Milad Shokouhi, Martin Szummer, and Nick Craswell. Lambdamerge: Merging the results of query reformulations. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 795–804, New York, NY, USA, 2011. ACM.

[161] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352, July 2006.

[162] Lixin Shi and Jian-Yun Nie. Using various term dependencies according to their utilities. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1493–1496, New York, NY, USA, 2010. ACM.

[163] Fabrizio Silvestri. Mining query logs: Turning search usage data into knowledge. *Found. Trends Inf. Retr.*, 4(1—2):1–174, January 2010.

[164] Amit Singhal and Fernando Pereira. Document expansion for speech retrieval. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 34–41, New York, NY, USA, 1999. ACM.

[165] Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809. Curran Associates, Inc., 2011.

[166] Ruihua Song, Michael J. Taylor, Ji-Rong Wen, Hsiao-Wuen Hon, and Yong Yu. Viewing term proximity from a different perspective. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR'08, pages 346–357. Springer-Verlag, Berlin, Heidelberg, 2008.

[167] Krysta M. Svore, Pallika H. Kanani, and Nazan Khan. How good is a span of terms?: exploiting proximity to improve web retrieval. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 154–161, New York, NY, USA, 2010. ACM.

[168] Bin Tan and Fuchun Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 347–356, New York, NY, USA, 2008. ACM.

[169] Tao Tao, Xuanhui Wang, Qiaozhu Mei, and ChengXiang Zhai. Language model information retrieval with document expansion. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 407–414, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[170] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 295–302, New York, NY, USA, 2007. ACM.

[171] Anastasios Tombros, Ian Ruthven, and Joemon M. Jose. How users assess web pages for information seeking. *J. Am. Soc. Inf. Sci. Technol.*, 56(4):327–344, February 2005.

[172] Kristina Toutanova and Robert C. Moore. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 144–151, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[173] Peter D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 491–502, London, UK, UK, 2001. Springer-Verlag.

[174] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 61–69, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[175] Chen Wang, Keping Bi, Yunhua Hu, Hang Li, and Guihong Cao. Extracting search-focused key n-grams for relevance ranking in web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 343–352, New York, NY, USA, 2012. ACM.

[176] Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 935–945. ACL, 2013.

[177] Jianqiang Wang and Douglas W. Oard. Matching meaning for cross-language information retrieval. *Inf. Process. Manage.*, 48(4):631–653, July 2012.

[178] Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 187–194, New York, NY, USA, 2009. ACM.

[179] Quan Wang, Zheng Cao, Jun Xu, and Hang Li. Group matrix factorization for scalable topic modeling. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 375–384, New York, NY, USA, 2012. ACM.

[180] Quan Wang, Jun Xu, Hang Li, and Nick Craswell. Regularized latent semantic indexing. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 685–694, New York, NY, USA, 2011. ACM.

[181] Quan Wang, Jun Xu, Hang Li, and Nick Craswell. Regularized latent semantic indexing: A new approach to large-scale topic modeling. *ACM Trans. Inf. Syst.*, 31(1):5:1–5:44, January 2013.

[182] Xuanhui Wang and ChengXiang Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 479–488, New York, NY, USA, 2008. ACM.

[183] Ziqi Wang, Gu Xu, Hang Li, and Ming Zhang. A fast and accurate method for approximate string search. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 52–61, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[184] Xing Wei and W. Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 178–185, New York, NY, USA, 2006. ACM.

[185] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 162–168, New York, NY, USA, 2001. ACM.

[186] Haocheng Wu, Yunhua Hu, Hang Li, and Enhong Chen. Query segmentation for relevance ranking in web search. *CoRR*, abs/1312.0182, 2013.

[187] Wei Wu, Hang Li, and Jun Xu. Learning query and document similarities from click-through bipartite graph with metadata. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 687–696, New York, NY, USA, 2013. ACM.

[188] Wei Wu, Zhengdong Lu, and Hang Li. Learning bilinear model for matching queries and documents. *J. Mach. Learn. Res.*, 14(1):2519–2548, January 2013.

[189] Wei Wu, Jun Xu, Hang Li, and Satoshi Oyama. Learning a robust relevance model for search using kernel methods. *J. Mach. Learn. Res.*, 12:1429–1458, July 2011.

[190] Gu Xu, Shuang-Hong Yang, and Hang Li. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1365–1374, New York, NY, USA, 2009. ACM.

[191] Jingfang Xu and Gu Xu. Learning similarity function for rare queries. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 615–624, New York, NY, USA, 2011. ACM.

[192] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '96, pages 4–11, New York, NY, USA, 1996. ACM.

[193] Jun Xu, Hang Li, and Chaoliang Zhong. Relevance ranking using kernels. In Pu-Jen Cheng, Min-Yen Kan, Wai Lam, and Preslav Nakov, editors, *Information Retrieval Technology*, volume 6458 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2010.

[194] Jun Xu, Wei Wu, Hang Li, and Gu Xu. A kernel approach to addressing term mismatch. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 153–154, New York, NY, USA, 2011. ACM.

[195] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 267–273, New York, NY, USA, 2003. ACM.

[196] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi, and WeiGuo Fan. Optimizing web search using web click-through data. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 118–126, New York, NY, USA, 2004. ACM.

[197] Xiaobing Xue, Yu Tao, Daxin Jiang, and Hang Li. Automatically mining question reformulation patterns from search log data. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 187–192, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[198] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 523–530, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

[199] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. Query by document. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 34–43, New York, NY, USA, 2009. ACM.

[200] Xing Yi and James Allan. A comparative study of utilizing topic models for information retrieval. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 29–41, Berlin, Heidelberg, 2009. Springer-Verlag.

[201] Xing Yi and James Allan. Discovering missing click-through query language information for web search. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 153–162, New York, NY, USA, 2011. ACM.

[202] Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 247–256, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[203] Fabio massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. A machine learning approach to textual entailment recognition. *Nat. Lang. Eng.*, 15(4):551–582, October 2009.

[204] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2(3):137–213, March 2008.

[205] Xian Zhang, Yu Hao, Xiaoyan Zhu, Ming Li, and David R. Cheriton. Information distance from a question to an answer. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 874–883, New York, NY, USA, 2007. ACM.

[206] Le Zhao and Jamie Callan. Term necessity prediction. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 259–268, New York, NY, USA, 2010. ACM.

[207] Le Zhao and Jamie Callan. Automatic term mismatch diagnosis for selective query expansion. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 515–524, New York, NY, USA, 2012. ACM.