Semistructured Data Search

Krisztian Balog

University of Stavanger
NO-4036 Stavanger, Norway
krisztian.balog@uis.no

Abstract. This paper presents a selection of methods for searching in heterogeneous data collections where some amount of structure is available. We start with a general retrieval framework, based on generative probabilistic modeling, for ranking unstructured document representations. Then, we consider structure at two different levels: documents and queries. For documents, the internal structure is captured through the use of multiple document fields, and various approaches to setting field weights are discussed. For queries, the focus is on effectively utilizing additional input data that the user might provide along with the keyword query, such as target categories or example documents. We place a particular emphasis on methods that are robust with respect to the availability of structured data and are able to deal with inconsistent or incomplete information.

Keywords: Semistructured data, generative probabilistic models, document modeling, query modeling.

1 Introduction

Traditionally, information retrieval (IR) systems have dealt with the problem of search in unstructured text collections. Database (DB) systems, on the other hand, have aimed at querying structured data that is highly organized and follows a strict schema. While this distinction still exists, it is not as sharp as a decade ago and some convergence between the two fields has occurred. To a large extent, this can be attributed to changes in the data landscape; over the past few years, alongside the document-oriented web, the Web of Data has emerged [8, 9]. This resulted in increased availability of (semi)structured data and made it possible to respond to users' queries with specific entities and objects, as opposed to merely a ranked list of documents. The Web has also changed users' expectations about how search applications should function; the single-search-box paradigm has become widespread, and ordinary users have little incentive to formulate structured queries (which would require the knowledge of the underlying schema as well as that of the query language). However, users might supply additional input data beyond the keyword query, such as target categories or example documents, provided that it is made sufficiently effortless for them to do so (for example, through specialized interfaces or query assistance services). This tutorial introduces a selection of methods that are able to effectively utilize structure that is present on the document side or arise on the query side.

N. Ferro (Ed.): PROMISE Winter School 2013, LNCS 8173, pp. 74–96, 2014.

1.1 Scope

For a long time, semistructured search was taken to be synonymous with XML retrieval. This paper offers a different perspective. Our unit of retrieval is documents, more precisely, document-based representations of entities or objects, which are assumed to be readily available. Unlike in XML retrieval, we do not provide responses beyond this level. Further, we assume that there is no separate schema; in XML, structure can be enforced by the XML schema with virtually the same rigor as in a relational database. Our approaches are primarily text-based; in Section 5.2 we introduce methods for modeling categories associated with documents, an idea that could be conceptualized further as an IR equivalent of categorical attribute values in databases. However, we do not support query operators, for example, for handling numerical values. Also, the methods we present in this chapter work with flat structures (but often with multiple ones); this stands in contrast with XML retrieval where much of the modeling efforts revolve around hierarchical structures. We particularly aim for methods that are robust with respect to the availability of structure and can be applied to a wide range of document types, from web documents written in HTML to entities described in RDF. Our general attitude towards query-side structure is that any additional input the user might provide beyond the keyword query is to be seen as complementary descriptions of the underlying information need and to be considered as "structural hints" as opposed to rigid formal constraints. For an excellent overview on XML retrieval we refer the reader to [26].

1.2 Organization

In Section 2 we explain what we mean by semistructured data search. Next, in Section 3 we present our general retrieval framework, based on generative probabilistic modeling techniques. This provides a common ground for methods that follow later and a principled way of accounting for the inherent uncertainty and heterogeneity involved with searching in this type of data. Section 4 starts with unstructured document retrieval and discusses methods for capturing internal document structure using multiple document fields. It also deals with questions related to setting field weights. Section 5 considers scenarios where the user can optionally complement the keyword query with additional information, such as target categories or example documents. Utilizing this extra input requires changes both on the query and on the document side. We conclude with a summary of our findings in Section 6.

2 Semistructured Data

In this section we briefly explain what we mean by *semistructured data* in the context of this tutorial. It is best understood in contrast to unstructured and structured data. The key characteristics are summarized in Table 1.

Unstructured data can be found in many different forms, including documents, spreadsheets, web pages, emails, blogs, tweets, and medical records. Generally

Table 1. Comparison of unstructured, semistructured, and structured data search

	Unstructured	**Semistructured**	**Structured**
Unit of retrieval	documents	objects	tuples
Schema	no	self-describing	fixed
Queries	keyword	keyword++	formal languages

speaking, it also includes non-textual data like images, video, and audio, however, our focus here is limited to textual data. There is little uniformity among the different forms, so content is utilized in an unstructured manner without making any further assumptions about the format. Retrieval in unstructured text collections is often referred to as *full-text search*.

Structured data is typically highly organized and tabular, such as the information stored in relational databases. The semantics of the data are captured in a data model and are mapped to a database schema. The schema describes various elements of the database, including tables, fields, and relationships, and imposes constraints to ensure the consistency of the data. Querying of the data is done using formal languages, like SQL.

Semistructured data is characterized by the lack of rigid, formal structure, such as the data models associated with relational databases. This means that there is no single schema to the data. Instead, the schema is contained within the data and is evolving together with the content, thereby making it "self-describing." Parts of the data yield little structure or lack structure altogether (e.g., plain text). Even when structural annotations are present, they are often ignored (e.g., full-text search). It is important to note that our take on semistructured data is somewhat different from the traditional view, especially when it comes to XML (cf. Section 1.1). Throughout this paper we will simply refer to the task of ranking documents. What we really mean by that is ranking document-based representations of objects or entities. Our queries are primarily keyword-based, which, optionally, can be complemented with additional components; hence, we refer to these as keyword++.

3 Retrieval Framework

The task we address is *ad-hoc retrieval*: answering a one-off query, representing the user's underlying information need, with a ranked list of documents. (Note that by documents we mean the document-based representations of objects or entities.) We approach this task in a generative probabilistic modeling framework. Generative models are attractive from both theoretical and empirical perspectives, and have been successfully applied to a wide range of retrieval problems [27, 44]. Importantly for us, generative models allow for a sound incorporation of structural clues into the retrieval model and are particularly well-suited for settings where training data is not available in large quantities. Another pragmatic consideration behind this choice is that it enables us to present the

material that follows in this chapter in a coherent and consistent manner. At the same time, we would like to emphasize that our main focus is on developing approaches for dealing with structure effectively and efficiently, and that generative models represent one possible solution for implementing general retrieval strategies; the same ideas may also be expressed in other retrieval frameworks.

Ranking documents given an input query q is done according to probability $P(d|q)$, computed for each document d in the collection. Instead of estimating this probability directly, a more accurate estimate may be obtained by applying Bayes' theorem:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \stackrel{\text{rank}}{=} P(q|d)P(d). \quad (1)$$

Notice that $P(q)$ in the denominator is the same for all documents, therefore, it is ignored for the purpose of ranking. We are then left with two main components: the *query likelihood* $P(q|d)$ and the *prior* probability of the document $P(d)$. Under this formulation the ranking of documents may be viewed as the following generative process. First, a particular document d is chosen with probability $P(d)$. Then, we subsequently attempt to draw the query q from this document with probability $P(q|d)$.

The simplest and most common way of estimating $P(q|d)$ is using multinomial unigram language models. Indeed, this is what we will be discussing first in the next section, then gradually moving to more complex estimation schemes based on document structure. In Section 5 we will move from keyword-only queries to semistructured queries containing semantic annotations, target types, or example documents. Making effective use of such additional information often necessitates query representations that go beyond the term level, breaking the query likelihood into multiple components. The prior probability of the document, $P(d)$, is often assumed to be uniform (and, as such, it can be ignored since it does not influence the ranking). Alternatively, it can be used encode query-independent evidence based on document length, authority, popularity, link structure, etc. [21, 22, 32].

4 Modeling Documents

This section is concerned with the modeling of internal document structure. We start with an unstructured "flat text" representation in Section 4.1. Then, we continue in Section 4.2 with an approach, which has been the predominant to date, to dealing with document structure: through the use of multiple document fields. Without exception, the models we discuss in this section are based on generative language modeling techniques. Language models, as the name suggests, represent language usage as statistical information associated with a vocabulary. A language model θ_d is built for each document d and then used to describe how likely this document would generate the query q, $P(q|\theta_d)$. We will cover this process in more detail next. Our presentation is self-contained, but the interested reader is referred to [45] for a full account on language modeling.

t	$P_{ML}(t\|d)$	t	$P_{ML}(t\|d)$
query	0.0150	language	0.0046
document	0.0135	field	0.0046
documents	0.0081	structure	0.0043
equation	0.0080	films	0.0036
data	0.0070	terms	0.0035
probability	0.0066	information	0.0035
model	0.0065	modeling	0.0033
retrieval	0.0058	example	0.0033
fields	0.0058	using	0.0031
term	0.0046	representation	0.0031

Fig. 1. Top terms with the corresponding term probabilities from the language model of this article (after stopword removal)

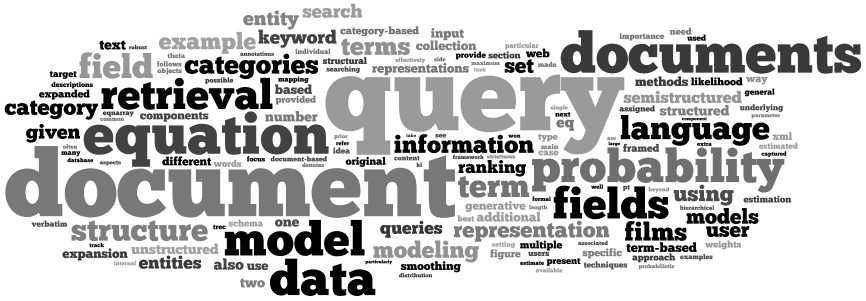

Fig. 2. Language model of this article, visualized as a tag cloud

4.1 Unstructured Document Representation

The simplest form of document representation is to ignore any structural clues or elements and take the whole document to be a bag of words. In this view of the document the exact ordering of terms is ignored and the importance of a term within the document is proportional to its number of occurrences (but is independent of where in the document those occurrences take place). Language models provide an elegant way of capturing this notion of term importance by representing each document as a multinomial probability distribution over the vocabulary of terms. We write θ_d to denote the model of document d, and the probability of a term t in the document's model is $P(t|\theta_d)$. This probability tells us how likely we would see t if we sampled a term randomly from d. Much of our efforts in this section will revolve around the estimation of this model.

The most straightforward way of obtaining the document language model is by using the *maximum likelihood* (ML) estimation:

$$P(t|\theta_d) = P_{ML}(t|d) = \frac{n(t,d)}{|d|}, \tag{2}$$

where $n(t,d)$ denotes the number of times term t occurs in document d and $|d|$ is the length of the document (i.e., $\sum_t n(t,d)$). This effectively means that the probability of the term equals to its relative frequency in the document. Figure 1 illustrates the idea of a document language model by listing the top terms (i.e., the ones with highest probabilities) given the language model of this article. A language model could be plotted as a histogram, like it is done in Figure 12, where bars correspond to terms and their heights are proportional to term probabilities. A visually more exciting alternative is to display it as a word cloud, such as the one shown in Figure 2. The document model θ_d is then used to estimate the probability of a given query q by taking the product of individual term probabilities as follows:

$$P(q|\theta_d) = \prod_{t \in q} P(t|\theta_d)^{n(t,q)}, \tag{3}$$

where $n(t,q)$ denotes the number of times term t is present in query q. Assuming uniform document priors (cf. Eq. 1) this probability can be used directly to produce a document ranking. To prevent numerical underflows, this computation, in practice, is performed in the log domain. We rewrite Eq. 3 as follows:

$$\log P(q|\theta_d) = \sum_{t \in q} n(t,q) \log P(t|\theta_d). \tag{4}$$

The document model constructed using the ML estimator has a severe limitation: unseen words in the document would get a zero probability. Moreover, because of the multiplication of individual term probabilities in Eq. 3, the whole query would be assigned a zero probability in such cases. Obviously, this is undesired behavior. The main purpose of *smoothing* is to assign a non-zero probability to the unseen words and to improve the accuracy of word probability estimation in general. This is typically achieved by discounting the probabilities of the words seen in the text and then assigning the extra probability mass to unseen words according to a background language model estimated using the entire collection:

$$P(t|\theta_d) = (1-\lambda)P_{ML}(t|d) + \lambda P_{ML}(t|C), \tag{5}$$

where the interpolation parameter λ controls the influence of the collection model, $P_{ML}(t|C)$, which is taken to be a maximum likelihood estimate:

$$P_{ML}(t|C) = \frac{\sum_d n(t,d)}{\sum_d |d|}. \tag{6}$$

This representation of the document as a mixture between the document and the collection is usually referred to as the *standard language modeling approach*.

The linear interpolation in Eq. 5 is also known as *Jelinek-Mercer smoothing*. Notice that the amount of smoothing applied is the same for all documents. Intuitively, longer documents may require less smoothing (as they already have a richer representation, through having more terms). The Bayes smoothing method, often referred to as *Dirichlet smoothing*, implements this idea by setting

$$\lambda = \frac{\mu}{|d| + \mu} \qquad (1 - \lambda) = \frac{|d|}{|d| + \mu}, \tag{7}$$

where μ is a parameter. Smoothing plays an important role in language modeling and the choice of the smoothing method and smoothing parameter can have a considerable impact on retrieval performance [47]. As a general rule of thumb, Dirichlet smoothing with μ set to the average document length in the collection is usually a reasonable starting point.

Despite its relative simplicity, the standard language modeling approach is a very powerful one; the overall idea of representing "things" by the language associated with them is intuitive and works well in many application scenarios.

4.2 Fielded Document Representation

Documents are rarely just flat text. For example, email messages have from, to, subject, and body fields; news articles are divided into title, lead, and body elements; web documents are annotated with structural markup using HTML. A common way to incorporate the internal structure of documents into the retrieval model is through the usage of *document fields*. Generally speaking, a field is made up of specific parts or segments of the document. We do not impose strict constraints on document fields. Specifically, we do not require each term occurrence to be assigned to exactly one field, i.e., fields can be overlapping. Also, fields do not necessarily have to cover the entire document, i.e., there may be parts of a document that are not associated with any specific field. In practical terms, fields usually correspond to the contents of particular markup tags provided by the structural annotation. In web document retrieval, for example, title, headings, meta keywords and descriptions, anchor text, and body text may be regarded as document fields [33]. Figure 3 shows an excerpt from a HTML file, with the corresponding field based document representation presented in Figure 4. When searching in the Web of Data, entities are described in the form of subject-predicate-object (SPO) triples of the RDF data model; a natural way of building a document-based representation of a particular entity is to consider all triples with the given subject, create separate fields for each predicate, and associate the corresponding object values with those fields; Figure 5 displays an example.

Next, we present an extension to our generative language modeling approach that makes us able to deal with multiple fields. Rather than using the language model estimated from a single document representation, this method estimates a mixture language model based on a combination of language models created from the various document fields [33]. We will refer to this approach as the *mixture of language models* (MLM).

```
<html>
   <head>
      <title>Winter School 2013</title>
      <meta name="keywords" content="PROMISE, school, PhD, IR, DB" />
      <meta name="description" content="PROMISE Winter School 2013" />
   </head>
   <body>
      <h1>PROMISE Winter School 2013</h1>
      <h2>Bridging between Information Retrieval and Databases</h2>
      <h3>Bressanone, Italy 4 - 8 February 2013</h3>
      <p>The aim of the PROMISE Winter School 2013 on "Bridging
      between Information Retrieval and Databases" is to give
      participants a grounding in the core topics that constitute the
      multidisciplinary area of information access and retrieval to
      unstructured, semistructured, and structured information.</p>
      [...]
   </body>
</html>
```

Fig. 3. Excerpt from a HTML page

Field	**Content**
title	Winter School 2013
meta	PROMISE, school, PhD, IR, DB PROMISE Winter School 2013
headings	PROMISE Winter School 2013 Bridging between Information Retrieval and Databases Bressanone, Italy 4 - 8 February 2013
body	The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information.

Fig. 4. Fielded document representation based on HTML markup for the document shown in Figure 3

Formally, let F be the set of possible fields, where $f \in F$ denotes a specific field. The document language model (θ_d) is taken to be a weighted linear combination of the field language models (θ_{d_f}):

$$P(t|\theta_d) = \sum_{f \in F} \alpha_f P(t|\theta_{d_f}), \tag{8}$$

where α_f is the weight associated with field f, such that $\sum_{f \in F} \alpha_f = 1$. The field-specific language models are estimated analogously to the unstructured

Field	Content
rdfs:label	Audi A4
rdfs:comment	The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the [...]
dbpprop:production	1994 2001 2005 2008
rdf:type	dbpedia-owl:MeanOfTransportation dbpedia-owl:Automobile
dbpedia-owl:manufacturer	dbpedia:Audi
dbpedia-owl:class	dbpedia:Compact_executive_car
owl:sameAs	freebase:Audi A4
is dbpedia-owl:predecessor of	dbpedia:Audi_A5
is dbpprop:similar of	dbpedia:Cadillac_BLS

Fig. 5. Excerpt from the fielded document representation of the entity *Audi A4*, based on its RDF description in DBpedia. URIs are shortened for convenience and are typeset in typewriter font. URLs in the content part are typically replaced by the name/label/title of the resource they point to [31, 32].

document model (cf. Eq. 5), the main differences being that term occurrences are restricted to the given field and a field-specific collection language model is used for smoothing:

$$P(t|\theta_{d_f}) = (1-\lambda_f)P_{ML}(t|d_f) + \lambda_f P_{ML}(t|C_f), \tag{9}$$

where both components are maximum likelihood estimates:

$$P_{ML}(t|d_f) = \frac{n(t,d_f)}{|d_f|} \qquad P_{ML}(t|C_f) = \frac{\sum_d n(t,d_f)}{\sum_d |d_f|} \tag{10}$$

In Eq. 10 $n(t,d_f)$ denotes the number of occurrences of term t in field f of document d and $|d_f|$ stands for the length of the field. The smoothing parameter λ_f is set using Dirichlet smoothing.

Now that we have discussed the retrieval model, two main questions remain to be addressed: (i) How to organize document content into fields? and (ii) How to estimate the corresponding field weights? As we shall see, these two questions are not independent of each other; having a larger number of fields can make the setting of field weights quite challenging. We differentiate between two settings. In one case, fields may be seen as alternative (in a sense "interchangeable") descriptions of the same content. In the other case, fields capture distinct properties or aspects; here, fields are characterized by distinctive term distributions. Our fundamental assumption, common to both settings, is that the information contained in the different fields is complementary in nature and that is why we would benefit from combining these fields.

Fields as Alternative Document Representations. A typical application scenario is web document retrieval. Fields used here include title, headings, meta

keywords and descriptions, anchor text, and body text. These are all descriptions of the same content and mainly differ in the number of words used (but not in the vocabulary). It is likely that many of the high probability terms according to the field language model (i.e., $P(t|\theta_{d_f})$) are shared among the different fields. Therefore, this approach rewards documents where the same query term appears in multiple fields.

In the absence of training data, field weights (α_f in Eq. 8) can be set uniformly (i.e., $\alpha_f = 1/|F|$, where $|F|$ is the total number of fields) or proportional to the field length (measured as the sum of field lengths of the given field type, i.e., $\alpha_f \propto \sum_d |d_f|$). An alternative solution is to set field weights proportional to their individual performance (that is, using only one particular field language model for ranking documents at a time) [33]. Obviously, this last method requires training queries with corresponding relevance assessments.

Fields Representing Distinct Aspects. Our use-case for illustrating this setting is entity retrieval, where documents describe a single type of entity, e.g., people or movies, and structured representations are readily available (for example in XML or RDF). For the sake of simplicity we assume that fields are not hierarchically organized. The underlying assumption here is that each query term has an implicit mapping to one or more fields (where "more" means at most a handful fields). Kim et al. [24] proposes a method, termed *probabilistic retrieval model for semi-structured data* (PRMS), that uses the distribution of words in the fields to provide clues for this mapping process. Under this approach the static field weights (α_f in Eq. 8) are replaced by a mapping probability $P(f|t)$:

$$P(t|\theta_d) = \sum_{f \in F} P(f|t) P(t|\theta_{d_f}). \tag{11}$$

The probability of mapping a (query) term t to a given document field f is estimated by applying Bayes' theorem and combining the prior field probability $P(f)$ and the probability of a term occurring in a given field $P(t|f)$:

$$P(f|t) = \frac{P(t|f)P(f)}{P(t)} = \frac{P(t|f)P(f)}{\sum_{f' \in F} P(t|f')P(f')}. \tag{12}$$

The prior probability of the term, $P(t)$, is further rewritten using the law of total probability (second step in Eq. 12); we write f' in the denominator when marginalizing over all possible fields so that to avoid confusion with the field f for which the mapping probability is being computed. In the end, we are left with two components to be estimated. The prior $P(f)$ is the probability of mapping the query term to field f before observing collection statistics; if could be set manually, for example, based on domain-specific background knowledge, or left to be uniform. The probability of a term given a field, $P(t|f)$ is conveniently estimated using the collection language model of that field, i.e., $P(t|f) = P_{ML}(t|C_f)$. This, we already have from earlier (see Eq. 10). Figure 6 shows the top fields and their mapping probabilities for an example query.

t = "Meg"		t = "Ryan"		t = "war"	
f	$P(f\|t)$	f	$P(f\|t)$	f	$P(f\|t)$
cast	0.407	cast	0.601	genre	0.927
team	0.381	team	0.381	title	0.070
title	0.187	title	0.017	location	0.002

Fig. 6. Example mapping probabilities for the query *Meg Ryan war* when searching in the IMDB collection

Two key assumptions made in PRMS are that (i) the collection is homogeneous and (ii) each field has a distinctive distribution of terms. These conditions are met in our example where the collection is limited to entities of a single type; in heterogenous collections with multiple types of entity, PRMS cannot be applied successfully [12]. One possible remedy is to rank each entity type with a type-specific model (that considers only fields specific to that type) and then merge results [23]. Another option is to reduce the number of fields considered by grouping them together [10, 31].

4.3 Further Reading

It was observed quite early in studies of retrieval models that searching on multiple document representations (such as title and abstract or free text and manually assigned index terms) and combining these representations during retrieval was more effective than searching on a single representation [13, 18]. All established retrieval frameworks have been generalized to multi-field document retrieval, including BM25 [38] and Divergence From Randomness [36]. These models, as well as the ones we have discussed earlier in this section, combine evidence from multiple fields on the term level, inside the document representation. It is also possible to combine evidence on the query level; the idea there is to rank documents using individual representations (possibly using different retrieval techniques depending on the particular representation) and then combine these retrieval results to produce a final ranking. This technique is often referred to as *meta-search* or *data fusion* in the literature [1, 30]. Robertson et al. [38] argue that components that contribute to the document score should be combined across fields at an earlier stage, i.e., on the term level and not on the query level. In this section we limited ourselves to flat structures, that is, a set of fields, ignoring any (hierarchical) relationships that may exist between them. Both language modeling and BM25 have been extended to handle hierarchical structures for element level XML retrieval [29, 34]. It is also possible to incorporate hierarchical field structures for entity retrieval, but the benefits of that over flat structures are yet to be explored [31].

5 Modeling Queries

A keyword query is a very sparse representation of the user's underlying information need. Obtaining a more detailed specification, a process known as

query modeling, has been a topic of active research from the very early years of IR [39]. The focus of our attention here is on queries that comprise not only a sequence of terms, but, optionally, additional components as well. It is paramount that we want to avoid the user having to use structured query languages. The non-keyword part that we aim for as extra input is (i) often highly application specific, (ii) typically collected through specialized user interfaces or query assistance services, and (iii) may or may not be provided by the user.[1] Such additional query components can entail, for example, (i) semantic annotations with entities or concepts, (ii) target types/categories, or (iii) examples of items (documents or entities) that the user deems relevant. This results in what we call a *keyword++* (or semistructured) query.

Previous benchmarking evaluation campaigns have presented several examples for scenarios that come with such enriched queries.[2] The TREC 2007 Enterprise track addresses a topic distillation task where users have to create overview pages on specific topics [3]. The additional information provided by the user consists of a small number of example documents; see Figure 7. The INEX 2007-2009 Entity Ranking track focuses on the retrieval of entities, where entities are represented by their corresponding Wikipedia article [14]. Keyword queries are complemented with target categories and/or a small number of example entities; see Figure 8. The TREC Entity track studies the related entity finding task: returning a ranked list of entities of a specified type that engage in a given relationship with a particular source entity [7]; see Figure 9.

Although this section is titled "modeling queries," in order to utilize this extra information, we will be required to make changes in the representation of documents as well, as we shall soon see.

5.1 Term-Based Modeling

The ranking of documents so far was based on (log) query likelihood, as defined in Eq. 4. We repeat this formula for convenience:

$$\log P(q|\theta_d) = \sum_{t \in q} n(t,q) \log P(t|\theta_d). \tag{13}$$

Throughout Section 4, our focus was on devising ways to improve the estimation of the document language model, $P(t|\theta_d)$. Next, we shift our attention to refining the representation of the query. Notice that this formula considers all query

[1] It is worth pointing out that the non-keyword components can also be obtained automatically; clearly this will not be of the same quality as if it was provided by the user explicitly, but can still improve retrieval performance. Importantly, estimating specific query components is a problem significantly easier than automatically translating an unstructured query into a structured one.

[2] The descriptions of information needs are called "topics" in TREC lingo. As can be seen in Figure 7 and 8, these also include narratives and/or extended descriptions of the information being sought. We do not use those fields here and consider only `<query>` in Figure 7 and `<title>` in Figure 8 as the keyword query.

```
<top>
   <num>CE-012</num>
   <query>cancer risk</query>
   <narr>
      Focus on genome damage and therefore cancer risk in humans.
   </narr>
   <page>CSIRO145-10349105</page>
   <page>CSIRO140-15970492</page>
   <page>CSIRO139-07037024</page>
   <page>CSIRO138-00801380</page>
</top>
```

Fig. 7. Example topic description from the TREC 2007 Enterprise track

terms with equal importance. What we would like, instead, is to be able to weigh individual terms differently. Therefore, we replace $n(t, q)$ with $P(t|\theta_q)$ and refer to it as the *query model*. Analogously to the document model, this is a probability distribution, i.e., $\sum_t P(t|\theta_q) = 1$. Substituting back to Eq. 13 we arrive at the following equation:

$$\log P(q|\theta_d) = \sum_{t \in q} P(t|\theta_q) \log P(t|\theta_d). \tag{14}$$

This generalized model equals to ranking based on negative Kullback-Leibler (KL) divergence between the query and document models,[3] also known as the *KL-divergence retrieval model* [25, 46]. The estimation of the document model θ_d is the same as with the query likelihood retrieval model, but the query model θ_q offers interesting opportunities for leveraging additional input and/or feedback information to improve retrieval accuracy.

In the baseline case, each query term receives equal weight:

$$P_{BL}(t|\theta_q) = \frac{n(t, q)}{|q|}, \tag{15}$$

where $|q|$ is the length of the query, measured in the number of terms. This is equivalent to the query likelihood scoring.

For improved query modeling, the basic idea is to interpolate the original (baseline) query model with an expanded query model $\hat{\theta}_q$:

$$P(t|\theta_q) = (1 - \alpha) P_{BL}(t|\theta_q) + \alpha P(t|\hat{\theta}_q), \tag{16}$$

where $\alpha \in [0, 1]$ is a parameter to control the importance of the expanded model. Using a mixture of the original and expanded query models ensures that we do not drift too far away from the original query. Figure 10 illustrates the idea with the original query shown on the left side and the expanded query (after mixing with the original query using Eq. 16) is on the right.

[3] Apart from a query-dependent constant, which does not affect the ranking.

```
<inex_topic topic_id="95" query_type="XER">
   <title>Tom Hanks movies where he plays a leading role.</title>
   <entities>
      <entity id="142417">Apollo 13</entity>
      <entity id="468293">Philadelphia</entity>
      <entity id="41528">Forrest Gump</entity>
      <entity id="158982">You've got mail</entity>
   </entities>
   <categories>
      <category id="101422">movies</category>
      <category id="81332">films</category>
   </categories>
   <description>
     This query should return the names of movies in which
     Tom Hanks played the leading role.
   </description>
   <narrative>
     Tom Hanks is a popular actor and the winner of many awards.
     This query should return his all the feature films in which he
     played the lead.
   </narrative>
</inex_topic>
```

Fig. 8. Example topic description from the INEX 2007 Entity Ranking track

```
<query>
   <num>22</num>
   <entity_name>
     Organization of Petroleum Exporting Countries (OPEC)
   </entity_name>
   <entity_homepage id="clueweb09-en0010-21-28880">
     http://www.opec.com/
   </entity_homepage>
   <target_entity>location</target_entity>
   <target_type_dbpedia>Country</target_type_dbpedia>
   <narrative>
     Find countries that are members of OPEC
     (the Organization of Petroleum Exporting Countries).
   </narrative>
 </query>
```

Fig. 9. Example topic description from the TREC 2011 Entity track

t	$P_{BL}(t\|\theta_q)$	t	$P(t\|\theta_q)$
machine	0.5000	vision	0.2796
vision	0.5000	machine	0.2762
		image	0.0248
		vehicles	0.0224
		safe	0.0220
		cam	0.0214
		traffic	0.0178
		technology	0.0176
		camera	0.0173
		object	0.0147

Fig. 10. Baseline (left) and expanded (right) query models for the query *machine vision*; only the top 10 terms are shown

One possible route for leveraging example documents (denoted as E), provided by the user as additional input, is to use them for estimating the expanded query model. This can be done with the help of (pseudo) relevance feedback techniques. We illustrate it with two popular and effective models, called *relevance models*, proposed in [28]. The principal idea is to construct relevance models based on co-occurrences of the original query terms with other terms in the set of feedback documents. We use these relevance models as our expanded query model $\hat{\theta}_q$. Relevance model 1 (RM1) assumes full independence between the original query terms and the expansion terms:

$$P_{RM1}(t|\hat{\theta}_q) \approx \sum_{d \in E} P(d) P(t|\theta_d) \prod_{t' \in q} P(t'|\theta_d), \tag{17}$$

where E is the set of example documents, and $P(t|\theta_d)$ is a smoothed document language model (cf. Eq. 5). Mind that t stands for a term in the expanded query model while $t' \in q$ denotes original query terms. For convenience, document priors are assumed to be uniform.

Relevance model 2 (RM2) tackles a different sampling strategy, where original query terms $t' \in q$ are still assumed to be independent of each other, but they are dependent on the expansion term t.

$$P_{RM2}(t|\hat{\theta}_q) \approx P(t) \prod_{t' \in q} \sum_{d \in E} P(t'|\theta_d) P(d|t), \tag{18}$$

where $P(d|t)$ is computed as

$$P(d|t) = \frac{P(t|\theta_d)P(d)}{P(t)} = \frac{P(t|\theta_d)P(d)}{\sum_{d' \in E} P(t|\theta_{d'})P(d')}. \tag{19}$$

In Eq. 19 the probability of a document given a term is first rewritten using Bayes' theorem, then the probability of the term in the denominator is marginalized over all example documents (denoted with d' to avoid confusion with d).

RM1 can be viewed as sampling of all query terms conditioned on t: a strong mutual independence assumption, compared to the pairwise independence assumptions made by RM2. Empirical evaluation has shown that RM2 is more robust, and performs slightly better that RM1 [4, 28].

The sampling of expansion terms does not have to be dependent on the original query. In a scenario where the user provides example documents, we can expect that these documents provide important aspects that are not covered by the keyword query. Thus, avoiding biasing the selection of expansion terms toward the original query can possibly lead to a more accurate representation of the underlying information need. The method introduced in [4] estimates the expanded query model as follows:

$$P_{EX}(t|\hat{\theta}_q) \approx \sum_{d \in E} P(t|\theta_d) P(d|E), \tag{20}$$

where $P(d|E)$ is the importance of a given document given the set of example documents E. In the simplest case, this probability is distributed uniformly among the examples, i.e., $P(d|E) = 1/|E|$. Alternatively, it is also possible to bias towards documents that are more relevant given the original query, $P(d|E) \propto P(d|q)$, or just the opposite—reward documents that are the most dissimilar to the query, assuming, that these bring in aspects that are not well described by the keyword query, $P(d|E) \propto 1 - P(d|q)$.

A word on pragmatic considerations before we move forward. The expanded query models tend to be quite large, as they contain all terms that appear in the feedback documents (the set of examples E in our case). Most of these expansion terms have an extremely small probability assigned to them and have negligible impact on document ranking, yet they slow down computation considerably. Therefore, it is common practice to limit expansion terms to the set of top 10-30 words with the highest term probability ($P(t|\hat{\theta}_q)$) as it provides a good tradeoff between effectiveness and efficiency [4, 43].

5.2 Category-Based Modeling

Next, we consider scenarios where the move beyond term-based representations, both for documents and for queries. We assume that a category system exists as part of the data collection and documents are assigned to one or more categories; a prime example for such data set is Wikipedia. As an illustration, Figure 11 lists the Wikipedia categories assigned to the article about the movie *Saving Private Ryan*. Moreover, we assume that the user provides a small number of target categories, along with the keyword query, like it is shown in Figure 8. In reality, category systems tend to grow quite large and are hierarchically organized. This presents a number of challenges. First, the categorization itself is imperfect; there might be inconsistencies, missing category assignments, documents placed in too general or too specific categories, and so on. Second, relevant results are not necessarily assigned to the categories provided by the user (who may not be completely familiar with the category system). Therefore, simply filtering on the target categories is insufficient, more robust techniques are needed.

```
1998 films | English-language films | 1990s drama films | 1990s war
films | Amblin Entertainment films | American war films | Best Drama
Picture Golden Globe winners | DreamWorks films | Epic films | Films
directed by Steven Spielberg | Films produced by Steven Spielberg |
Films set in France | Films set in 1944 | Films shot in the Republic
of Ireland | Films that won the Best Sound Mixing Academy Award |
Films whose cinematographer won the Best Cinematography Academy Award
| Films whose director won the Best Director Academy Award | Films
whose director won the Best Director Golden Globe | Films whose editor
won the Best Film Editing Academy Award | Operation Overlord films |
Paramount Pictures films | War epic films | War films
```

Fig. 11. Wikipedia categories for the movie *Saving Private Ryan*

A standard way of using category information is to include a separate category similarity component in the overall ranking formula [6, 16, 22, 35]. A principled way realizing this idea is proposed in [6], where both documents and queries have a dual representation, one based on terms and one based categories. Formally, each document is represented as a pair, $d = (\theta_d^T, \theta_d^C)$, where θ_d^T is a probability distribution over terms and θ_d^C is a probability distribution over categories. Similarly, the query is also represented as a pair, $q = (\theta_q^T, \theta_q^C)$. The overall idea is illustrated in Figure 12, where the left and right sides symbolize the query and the document, respectively, where each have a term-based (top) and a category-based (bottom) representation, modeled as multinomial probability distributions. A word on notation before we continue: the type of representation, term-based or category-based, is indicated with T or C in the superscript; q and d in the subscript stand for query and document, respectively.

The probability of a document generating the query is estimated using a mixture of term-based and category-based components:

$$P(q|d) = \lambda_t \cdot P(\theta_q^T|\theta_d^T) + (1 - \lambda_t) \cdot P(\theta_q^C|\theta_d^C), \tag{21}$$

where λ_t is an interpolation parameter. Notice that the term-based component in Eq. 21 is essentially what we have worked on so far, and could be computed using Eq. 14. However, due to pragmatic reasons (see below) we need to include an additional transformation step. For the sake of space considerations, we detail only the term-based component. The category-based component is computed analogously (specifically, by replacing t with c and T with C in Eqs. 22, 23, and 24). We use KL divergence as the basis of distributional similarity:

$$KL(\theta_q^T||\theta_d^T) = \sum_t P(t|\theta_q^T) \cdot \log \frac{P(t|\theta_q^T)}{P(t|\theta_d^T)}. \tag{22}$$

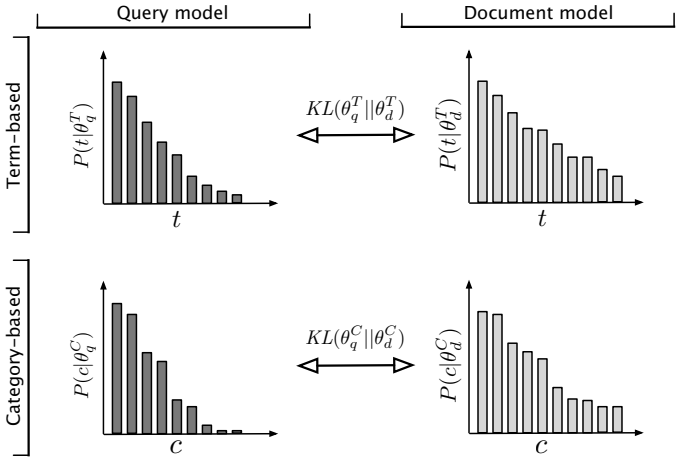

Fig. 12. A two-component model where both terms and categories are represented as probability distributions (top vs. bottom) for both queries and documents (left vs. right). KL divergence is used as the basis of distributional similarity.

Since KL divergence is a score (which is lower when two distributions are more similar), we turn it into a probability using Eq. 23:

$$P(\theta_q^T|\theta_d^T) = z^T \cdot \big(\max KL(\theta_q^T||\cdot) - KL(\theta_q^T||\theta_d^T)\big), \tag{23}$$

where $\max KL(\theta_q^T||\cdot)$ is the maximum KL divergence score observed for query q. Further, z^T is a normalization factor set as follows:

$$z^T = 1/\sum_d \max\big(KL(\theta_q^T||\cdot) - KL(\theta_q^T||\theta_d^T)\big). \tag{24}$$

Observe that Eq. 23 ranks documents in the same order as Eq. 14 does, albeit they differ in the actual values assigned. This does not make a difference when a single representation is used, but becomes an issue when term-based and category-based components need to be combined. Simply put, this transformation ensures that the probabilities are computationally tractable and "compatible;" the interested reader is referred to [6] for further details.

What remains to be defined, then, is the estimation of the ingredients for category-based similarity. Specifically, we need to define the probability of a category given a query, $P(c|\theta_q^C)$, and the probability of a category given a document, $P(c|\theta_d^C)$. We start with the latter. Similarly to the term-based representation, we need to employ smoothing on the document side to ensure that $P(c|\theta_d^C) > 0$ for all categories that might appear in the query. Analogously to the term-based case, we smooth the maximum likelihood estimate with a background model:

$$P(c|\theta_d^C) = (1 - \lambda^C)P_{ML}(c|d) + \lambda P_{ML}(c|C), \tag{25}$$

where the interpolation parameter λ^C controls the influence of the collection model, and can be set using guidance from Dirichlet prior smoothing (following the intuition that documents with a richer category-based representation require less smoothing). Further, we set

$$P_{ML}(c|d) = \frac{n(c,d)}{\sum_{c'} n(c',d)} \qquad P_{ML}(c|C) = \frac{\sum_d n(c,d)}{\sum_{c'} \sum_d n(c',d)}, \tag{26}$$

where $n(c,d)$ is 1 if category c is assigned to document d and 0 otherwise.

As for the probability of a category given a query, $P(c|\theta_q^C)$, we have a number of options. The baseline approach is to use the categories provided by the user and assign the same importance to each:

$$P_{BL}(c|\theta_q^C) = \frac{n(c,q)}{\sum_{c'} n(c',q)}, \tag{27}$$

where $n(c,q)$ is 1 if category c is present in the query and 0 otherwise. It is also possible to use the keyword query to obtain a ranking of categories (based on category labels or the contents of documents that belong to each category) and take the top-ranked categories (either with equal importance or with weights set proportional to the retrieval scores). This strategy has shown to be beneficial in expanding the typically small set of input categories provided by the user. It is worth pointing out that this method is applicable even when no input category information is given by the user at all.

A particularly nice feature of this framework is that it allows for category-based expansion analogously to the term-based case. Techniques introduced in Section 5.1 (blind relevance feedback and sampling from example documents) can be adopted in a straightforward way to categories. In fact, it has been shown that category-based feedback can be more beneficial than term-based feedback [6]. There exist further possibilities specific to categories. Most notably, hierarchical relationships between categories can also be utilized for expansion, for example, by considering parent and/or sub-categories up to a certain depth [15, 20, 42, 48].

5.3 Further Reading

Query expansion techniques mostly fall into two main categories: global and local. The idea of *global* analysis is to expand the query using global collection statistics based, for instance, on a co-occurrence analysis of the entire collection or using domain specific background knowledge [2]. *Local* approaches, on the other hand, typically use (known or assumed-to-be) relevant documents as examples from which expansion terms are selected [39, 41]; the methods we presented in Section 5.1 fall into this category. For a comprehensive overview on query expansion methods we refer the reader to [11]. Category-based modeling, discussed in Section 5.2, can be seen as a variant of *concept-based* information retrieval, where both documents and queries are represented using semantic concepts, instead of or in addition to keywords [17]. The TREC Entity track presents a scenario where the query seeks to find related entities ("Airlines that

currently use Boeing 747 planes") and is annotated with the input entity ("Boeing 747") and with the target type ("Airline") [5]. Later editions of the track anchored these annotations in a knowledge base (DBpedia) [7]. Obtaining such semantic annotations for keyword queries automatically is a topic of active research, often involving methods at the intersection of information retrieval and databases [37, 40].

6 Summary

In this paper we have looked at various ways of utilizing structure for improving the ranking of document-based representations of objects or entities. The internal structure of documents can effectively be captured through the use of multiple document fields. Depending on the data source and application, these fields might be alternative descriptions of the same content or record different aspects of it; the two call for different field weight estimation methods. Structure, to a certain degree, can also be captured on the query side, even without the use of formal query languages. Users, for example, can provide target categories or a few example documents, if it is made sufficiently effortless for them through specialized interfaces or query assistance services. This extra information can then be used to obtain a richer representation of the underlying information need in the form of expanded query models. Finally, both documents and queries can be modeled beyond the term space. We have illustrated this using categories; this assumes a setting where documents are classified according to some category system and the user might supplement the keyword query with a small number of target categories. A particularly nice property of the proposed model is that query expansion techniques developed for the term-based representation can be adopted in a straightforward way to categories—this provides an effective solution to handle noisy category information.

References

[1] Aslam, J.A., Montague, M.: Models for metasearch. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001), pp. 276–284. ACM (2001)
[2] Bai, J., Nie, J.-Y.: Adapting information retrieval to query contexts. Inf. Process. Manage. 44(6), 1901–1922 (2008)
[3] Bailey, P., Craswell, N., de Vries, A.P., Soboroff, I.: Overview of the TREC 2007 enterprise track. In: The Sixteenth Text REtrieval Conference Proceedings (TREC 2007). NIST Special Publication 500-274 (2008)
[4] Balog, K., Weerkamp, W., de Rijke, M.: A few examples go a long way: constructing query models from elaborate query formulations. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 371–378. ACM (2008)
[5] Balog, K., de Vries, A.P., Serdyukov, P., Thomas, P., Westerveld, T.: Overview of the TREC 2009 entity track. In: Proceedings of the Eighteenth Text REtrieval Conference (TREC 2009), NIST Special Publication 500-278 (February 2010)

[6] Balog, K., Bron, M., De Rijke, M.: Query modeling for entity search based on terms, categories, and examples. ACM Trans. Inf. Syst. 29(4), 22:1–22:31 (2011)
[7] Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the TREC 2011 entity track. In: The Twentieth Text REtrieval Conference Proceedings (TREC 2011). NIST Special Publication 500-296 (February 2012)
[8] Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
[9] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. Web Semant. 7(3), 154–165 (2009)
[10] Blanco, R., Mika, P., Vigna, S.: Effective and efficient entity search in RDF data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 83–97. Springer, Heidelberg (2011)
[11] Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Comput. Surv. 44(1), 1:1–1:50 (2012)
[12] Dalton, J., Huston, S.: Semantic entity retrieval using web queries over structured RDF data. In: Proceedings of the 3rd International Semantic Search Workshop, SEMSEARCH 2010 (2010)
[13] Das-Gupta, P., Katzer, J.: A study of the overlap among document representations. In: Proceedings of the 6th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1983), pp. 106–114. ACM (1983)
[14] de Vries, A.P., Vercoustre, A.-M., Thom, J.A., Craswell, N., Lalmas, M.: Overview of the INEX 2007 entity ranking track. In: Fuhr, et al. (eds.) [19], pp. 245–251
[15] Demartini, G., Firan, C.S., Iofciu, T.: L3S at INEX 2007: Query expansion for entity ranking using a highly accurate ontology. In: Fuhr, et al. (eds.) [19], pp. 252–263
[16] Demartini, G., Firan, C.S., Iofciu, T., Krestel, R., Nejdl, W.: Why finding entities in Wikipedia is difficult, sometimes. Inf. Retr. 13(5), 534–567 (2010)
[17] Egozi, O., Markovitch, S., Gabrilovich, E.: Concept-based information retrieval using explicit semantic analysis. ACM Trans. Inf. Syst. 29(2), 8:1–8:34 (2011)
[18] Fisher, H.L., Elchesen, D.R.: Effectiveness of combining title words and index terms in machine retrieval searches. Nature 238, 109–110 (1972)
[19] Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.): INEX 2007. LNCS, vol. 4862. Springer, Heidelberg (2008)
[20] Jämsen, J., Näppilä, T., Arvola, P.: Entity ranking based on category expansion. In: Fuhr, et al. (eds.) [19], pp. 264–278
[21] Kamps, J., Mishne, G., de Rijke, M.: Language models for searching in Web corpora. In: The Thirteenth Text REtrieval Conference Proceedings (TREC 2004). NIST Special Publication 500-261 (2005)
[22] Kaptein, R., Kamps, J.: Exploiting the category structure of Wikipedia for entity ranking. Artif. Intell. 194, 111–129 (2013)
[23] Kim, J., Croft, W.B.: Ranking using multiple document types in desktop search. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010), pp. 50–57. ACM (2010)
[24] Kim, J., Xue, X., Croft, W.B.: A probabilistic retrieval model for semistructured data. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 228–239. Springer, Heidelberg (2009)

[25] Lafferty, J., Zhai, C.: Document language models, query models, and risk minimization for information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001), pp. 111–119. ACM (2001)

[26] Lalmas, M., Baeza-Yates, R.: Structured text retrieval. In: Modern Information Retrieval - The Concepts and Technology Behind Search, 2nd edn. Pearson Education Ltd., Harlow (2011)

[27] Lavrenko, V.: A Generative Theory of Relevance. The Information Retrieval Series, vol. 26. Springer, Heidelberg (2008)

[28] Lavrenko, V., Croft, W.B.: Relevance based language models. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001), pp. 120–127. ACM (2001)

[29] Lu, W., Robertson, S., MacFarlane, A.: Field-weighted XML retrieval based on BM25. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 161–171. Springer, Heidelberg (2006)

[30] Montague, M., Aslam, J.A.: Condorcet fusion for improved retrieval. In: Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM 2002), pp. 538–548. ACM (2002)

[31] Neumayer, R., Balog, K., Nørvåg, K.: On the modeling of entities for ad-hoc entity search in the web of data. In: Baeza-Yates, R., de Vries, A.P., Zaragoza, H., Cambazoglu, B.B., Murdock, V., Lempel, R., Silvestri, F. (eds.) ECIR 2012. LNCS, vol. 7224, pp. 133–145. Springer, Heidelberg (2012)

[32] Neumayer, R., Balog, K., Nørvåg, K.: When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In: Baeza-Yates, R., de Vries, A.P., Zaragoza, H., Cambazoglu, B.B., Murdock, V., Lempel, R., Silvestri, F. (eds.) ECIR 2012. LNCS, vol. 7224, pp. 540–543. Springer, Heidelberg (2012)

[33] Ogilvie, P., Callan, J.: Combining document representations for known-item search. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003), pp. 143–150. ACM (2003)

[34] Ogilvie, P., Callan, J.: Hierarchical language models for XML component retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 224–237. Springer, Heidelberg (2005)

[35] Pehcevski, J., Thom, J.A., Vercoustre, A.-M., Naumovski, V.: Entity ranking in Wikipedia: utilising categories, links and topic difficulty prediction. Inf. Retr. 13(5), 568–600 (2010)

[36] Plachouras, V., Ounis, I.: Multinomial randomness models for retrieval with document fields. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECIR 2007. LNCS, vol. 4425, pp. 28–39. Springer, Heidelberg (2007)

[37] Pound, J., Hudek, A.K., Ilyas, I.F., Weddell, G.: Interpreting keyword queries over web knowledge bases. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012), pp. 305–314. ACM (2012)

[38] Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 extension to multiple weighted fields. In: Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM 2004), pp. 42–49. ACM (2004)

[39] Rocchio, J.J.: Relevance feedback in information retrieval. In: Salton, G. (ed.) The SMART Retrieval System: Experiments in Automatic Document Processing, pp. 313–323. Prentice-Hall, Inc. (1971)

[40] Sawant, U., Chakrabarti, S.: Learning joint query interpretation and response ranking. In: Proceedings of the 22nd International Conference on World Wide Web (WWW 2013), pp. 1099–1110. International World Wide Web Conferences Steering Committee (2013)
[41] Tao, T., Zhai, C.: Regularized estimation of mixture models for robust pseudo-relevance feedback. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), pp. 162–169. ACM (2006)
[42] Thom, J., Pehcevski, J., Vercoustre, A.-M.: Use of Wikipedia categories in entity ranking. In: The 12th Australasian Document Computing Symposium, ADCS 2007 (2007)
[43] Weerkamp, W., Balog, K., de Rijke, M.: Exploiting external collections for query expansion. ACM Trans. Web 6(4), 18:1–18:29 (2012)
[44] Westerveld, T., Vries, A., Jong, F.: Generative probabilistic models. In: Blanken, H.M., Blok, H.E., Feng, L., Vries, A.P. (eds.) Multimedia Retrieval. Data-Centric Systems and Applications, pp. 177–198. Springer, Heidelberg (2007)
[45] Zhai, C.: Statistical language models for information retrieval: a critical review. Found. Trends Inf. Retr. 2, 137–213 (2008)
[46] Zhai, C., Lafferty, J.: Model-based feedback in the language modeling approach to information retrieval. In: Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001), pp. 403–410. ACM (2001)
[47] Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. 22, 179–214 (2004)
[48] Zhu, J., Song, D., Rüger, S.: Integrating document features for entity ranking. In: Fuhr, et al. (eds.) [19], pp. 336–347