Entity Retrieval via Entity Factoid Hierarchy[*]

Chunliang Lu, Wai Lam, Yi Liao
Key Laboratory of High Confidence Software Technologies
Ministry of Education (CUHK Sub-Lab)
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
{cllu,wlam,yliao}@se.cuhk.edu.hk

Abstract

We propose that entity queries are generated via a two-step process: users first select entity facts that can distinguish target entities from the others; and then choose words to describe each selected fact. Based on this query generation paradigm, we propose a new entity representation model named as entity factoid hierarchy. An entity factoid hierarchy is a tree structure composed of factoid nodes. A factoid node describes one or more facts about the entity in different information granularities. The entity factoid hierarchy is constructed via a factor graph model, and the inference on the factor graph is achieved by a modified variant of Multiple-try Metropolis algorithm. Entity retrieval is performed by decomposing entity queries and computing the query likelihood on the entity factoid hierarchy. Using an array of benchmark datasets, we demonstrate that our proposed framework significantly improves the retrieval performance over existing models.

1 Introduction

Entity retrieval, which aims at returning specific entities to directly answer a user's query, has drawn much attention these years. Various entity retrieval tasks have been proposed, such as TREC Entity (Balog et al., 2012; Wang et al., 2011) and INEX-LD (Wang et al., 2012; Wang and Kang, 2012). Many existing entity retrieval models follow the document retrieval assumption: when issuing queries, users choose the words that may appear in the "entity pseudo-document". Based on the assumption, these models construct internal entity representations by combining various entity descriptions, and use these representations to compute the rank of the candidate entities for a given entity query. These models include fielded versions of BM25 and Mixture of Language Models (Neumayer et al., 2012), Entity Language Model (Raghavan et al., 2004), Hierarchical Expert Model (Petkova and Croft, 2006), Structured Positional Entity Language Model (Lu et al., 2013).

However, a closer examination of entity queries reveals that most of them are not simple uniform word samples from the "entity pseudo-document". Instead, they can be decomposed into multiple parts, where each part describes a fact about target entities. For example, the query "National capitals situated on islands" describes two facts regarding a target entity: it is a national capital; it is located on an island. Compared to the assumption in document retrieval models, where query terms are assumed to be generated from a single document, these query terms can be regarded to be independently generated from two underlying documents. According to this observation, we propose that an entity query is generated via a two-step process: users first select facts that can distinguish target entities from the others; and then choose words that describe the selected facts. Based on the proposed query generation paradigm, we design a new entity retrieval framework. On one hand, an entity is modeled to have multiple internal representations, each regarding one or more closely related facts. On the other hand, an entity query is decomposed into one or more subqueries, each describing a fact about target entities. In this way, entity retrieval can be performed by combining the probabilities of subqueries being satisfied for each candidate entity.

One of the central components of our proposed

[*] The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: 413510 and 14203414) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055034).

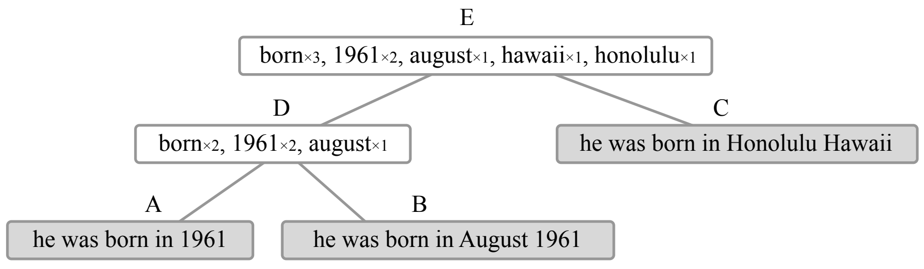
E
born×3, 1961×2, august×1, hawaii×1, honolulu×1
D
born×2, 1961×2, august×1
C
he was born in Honolulu Hawaii
A
he was born in 1961
B
he was born in August 1961

Figure 1: An example of entity factoid hierarchy containing two factoids about Barack Obama

retrieval framework is a novel entity representation known as entity factoid hierarchy. An entity factoid hierarchy is a tree structure composed of factoid nodes, which is automatically constructed from a collection of entity descriptions. We abuse the term "factoid" to denote a single piece of information regarding an entity. A factoid node in the hierarchy describes one or more factoids. Factoid nodes in different levels capture the information of different levels of detail (referred to as information granularities hereafter), where lower level nodes contain more detailed information and higher level nodes abstract the details away. The entity factoid hierarchy is constructed via a factor graph model, and the inference on the factor graph is achieved by a modified variant of Multiple-try Metropolis algorithm. Each factoid node is indexed separately as a pseudo-document. During retrieval, the query likelihood for a candidate entity are computed by transversing the factoid hierarchy. Compared to exiting entity retrieval models, our proposed framework exhibits two advantages:

- By organizing entity descriptions in a hierarchical structure, detailed entity information is preserved and we can return finer confidence value. Suppose that the entity "Barack Obama" is only described by one sentence: "born in 1961". Traditional entity models, which model an entity as a pseudo-document, would return high confidence value for the query "who is born in 1961". However, as we add more and more sentences to describe "Barack Obama", the confidence value returned for the query decreases due to the longer entity pseudo-document. This result is not desirable for entity retrieval, since adding more descriptions about other facts should not affect the confidence of existing facts. Our factoid hierarchy avoids this problem by preserving all the entity descriptions in a hierarchical structure. When performing retrieval, entity factoid hierarchy can be traversed to locate the best supporting description for the query.

- By separating entity facts in different factoid nodes, our model prevent ambiguity caused by mixing terms describing different facts. Suppose "Barack Obama" is described by two sentences: "Barack Obama is a president of United States" and "Barack Obama is a graduate of Harvard Law School", and our query is "Who is a president of Harvard Law School". A traditional document retrieval model with a bag-of-word entity pseudo-document would return "Barack Obama" with high confidence, since all the query terms appear in the entity descriptions. But obviously, this result is not correct. In our factoid hierarchy, these two facts are separated in lower level factoid nodes. While higher level nodes are still mixed with terms from child nodes, they are penalized to avoid giving high confidence value.

2 Factoid Hierarchy

2.1 Hierarchy Representation

As mentioned in the previous section, all the information regarding an entity is organized in a particular factoid hierarchy. We denote the term "factoid" as a single piece of information regarding an entity, such as the birth date of Barack Obama. A factoid node in the hierarchy describes one or more factoids. Each factoid node is associated with a bag-of-words vector to represent the factoid description. Factoid nodes in different depth encode information in different granularities.

An example of an entity factoid hierarchy, regarding two factoids (birth date and birth place) about Barack Obama, is given in Figure 1. The example hierarchy is constructed from three sentences about Barack Obama: he was born in 1961; he was born in August 1961; he was born in Honolulu Hawaii. These three sentences correspond to the leaf nodes A, B, and C respectively in Figure 1. In general, a leaf node in the factoid hierarchy comes directly from a sentence or a RDF triple describing the entity. Since it is extracted either from human written texts or from manually crafted structured databases, a leaf node represents the most exact representation regarding one or more factoids. During the construction of the hierarchy, intermediate nodes are formed as parents for nodes that contain closely related factoids. The factoid description for an intermediate node is the sum of bag-of-words vectors of its child nodes. In this way, intermediate nodes capture the words that are used more frequently with higher weights to describe the underlying factoids in a more general form. As we merge more nodes and move up in the hierarchy, intermediate nodes become blended with more different factoids. Node D in Figure 1 is an intermediate factoid node, as a parent node for nodes A and B both describing the birth date. The root node in an entity factoid hierarchy summarizes all the descriptions regarding an entity, which is similar to the "entity pseudo-document" used in some existing entity retrieval models. Each entity factoid hierarchy has only one root node. For example, node E in Figure 1 is the root node, and it contains words from all the three sentences.

Note that the depth of a leaf node varies with the number of descriptions associated with the factoids. Some factoids may be associated with lots of detailed information and are expressed in many sentences, while others are only expressed in one or two sentences. For example, the factoid that Obama is elected president in 2008 may be described in many sentences and in different contexts; while the factoid that Obama is born in Kapiolani Maternity & Gynecological Hospital is only mentioned in a few sentences. In this case, factoid nodes associated with more details may have deeper hierarchical structure.

2.2 Factor Graph Model

To construct the entity factoid hierarchy, we make use of a hierarchical discriminative factor graph model. A similar factor graph model has been proposed to solve the coreference resolution in (Singh et al., 2011; Wick et al., 2012). Here we design a factor graph model corresponding to the entity factoid hierarchy, together with new factor types and inference mechanism.

Generally speaking, a factor graph is composed of two parts: a set of random variables and a set of factors that model the dependencies between random variables. An example of the factor graph construction corresponding to the factoid hierarchy involved in Figure 1 is given in Figure 2. In our factor graph approach, each factoid is represented as a random variable f_i, corresponding to a rounded square node in Figure 2. The pairwise binary decision variable y_{ij}, denotes whether a factoid f_i is a child of another factoid f_j corresponding to a circle node in Figure 2. The set of factoids F plus the set of decision variables $\mathbf{y}$ are the random variables in our factor graph model. To model the dependency between factoids, we consider two types of factors. Ψ_p is the set of factors that consider the compatibility between two factoid nodes, i.e., to indicate whether two nodes have parent-child relationship. Ψ_u is the set of factors that measure the compatibility of the factoid node itself. Such factor is used to check whether a new intermediate node should be created. Factors are represented as square nodes in Figure 2. Given a factor graph model $\mathbf{m}$, our target is to find the best assignments for the decision variable $\mathbf{y}$ that maximizes the objective function in Equation (1).

$$P(\mathbf{y}, F|\mathbf{m}) = \prod_{f \in \mathcal{F}} \Psi_p(f, f^p)\Psi_u(f) \qquad (1)$$

2.3 Factors Design

The pairwise factors Ψ_p and unit-wise factors Ψ_u compute the compatibility scores among factoid nodes. Each factor type is associated with a weight w to indicate the importance of the factor during inference. For the notation, the bag-of-words representation for a factoid node is denoted as d. We use superscripts p and c to denote the variables of parent nodes and child nodes. To capture the interrelations between factoid nodes, the following factors are used in our factor graph model.

Bag-of-words similarity To check whether two factoid nodes refer to the same fact, we compare the similarity between their bag-of-words descriptions. We choose Kullback-Leibler divergence (KL divergence) as the similarity measure. By definition, the KL divergence of Q from P, denoted $D_{KL}(P||Q)$, is a measure of the informa-

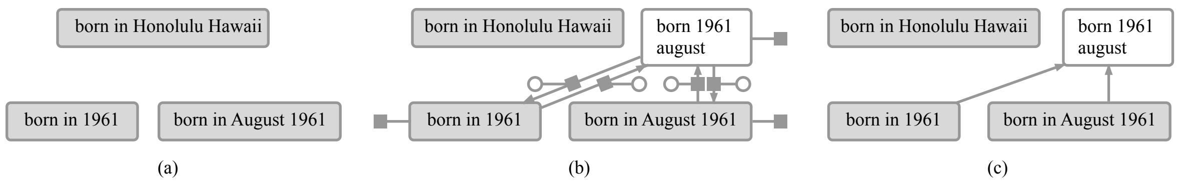

Figure 2: Generation of an factoid hierarchy via factor graph inference. Factoid nodes are initialized as singletons in (a). During one step of sampling in (b), two factoid nodes are selected and one proposal is to add a common parent. If we accept the proposal, we end up with the factoid hierarchy in (c).

tion lost when Q is used to approximate P. It is a non-symmetric measure and fits in our problem nicely, i.e., measuring whether a parent node is a more abstract representation of its child node. The compatibility score is computed as:

$$\begin{aligned} &- w_1 \cdot D_{KL}(d^p || d^q) \\ =& - w_1 \cdot \sum_{i=1}^{m} d_i^p \times \log\left(\frac{d_i^p}{d_i^c}\right), \end{aligned} \tag{2}$$

where d_i^p is the smoothed term frequency of the factoid description for the parent node; d_i^c is for the child node; w_1 is a global weighting parameter among different factors. In fact, we have also explored other popular text similarity metrics summarized in (Huang, 2008), and find that KL divergence performs the best.

Entropy penalty We penalize the entropy of the factoid description to encourage a smaller vocabulary of words describing the underlying factoids:

$$-w_2 \cdot \frac{H(d)}{\log ||d||_0}, \tag{3}$$

where $H(d)$ denotes the Shannon entropy for the bag-of-words representation of the factoid description d; $||d||_0$ is the number of unique terms in the factoid description.

Structure penalty The depth of a factoid node indicates the level of information granularity. However, we also need to control the depth of the factoid hierarchy. A factoid node should not have too many levels. We define the depth penalty as:

$$-w_3 \cdot |n_d - \frac{||d||_0}{s}|, \tag{4}$$

where n_d is the depth of a factoid node and s is the parameter that controls the average depth of factoid nodes per term. In this way, we can control the average depth of factoid nodes in the entity factoid hierarchy.

2.4 Inference

Exact inference is impossible for our factor graph model due to the large state space. Here we adopt a modified variant of Multiple-try Metropolis algorithm to conduct maximum probability estimation for inference, following the work in (Wick et al., 2013). At each sampling step, multiple changes to the current setting are proposed. The acceptance probability for a given proposal is equal to the likelihood ratio of the proposed hypothesis to the current hypothesis. In our case, we initialize the MCMC procedure to the singleton configuration, where each entity description, such as a sentence or a RDF triple, forms its own factoid hierarchy initially. At each sampling step, we randomly select two nodes and propose several alternative local modifications. If f_i and f_j are not connected, i.e., sharing no common child nodes, the following changes are proposed:

- Add factoid f_i as the parent of f_j, if f_j has no parent node;
- Remove f_j from its current parent, if f_j has a parent;
- Create a new common parent for f_i and f_j, if both f_i and f_j have no parent.

Otherwise, if f_i and f_j are in the same cluster, the following changes are proposed:

- Remove f_j from its current parent;
- Move f_j's children to f_j's parent and delete f_j, if f_j is an intermediate node.

A sampling step of the inference process is illustrated in Figure 2. Initially, all the decision variables $\mathbf{y}$ are set to zero. That is, each factoid node is regarded as forming its own factoid hierarchy, as illustrated in Figure 2(a). During the inference, local modifications are proposed to the current factor graph hypothesis. For example, in Figure 2(b), the two factoid nodes at the bottom are selected and proposed to add a new intermediate factoid as their common parent. If we accept the proposal, we get an intermediate factoid hierarchy as illustrated in Figure 2(c).

The sampling process is iterated until no proposal has been accepted in a certain number of successive steps, or a maximum number of steps has been reached. Each entity factoid hierarchy is inferred separately, allowing us to parallelize the inference across multiple machines.

3 Entity Retrieval

3.1 Retrieval Model

After we preprocess available information sources and construct the entity factoid hierarchy, we are ready to answer entity queries. Our retrieval model is based on the query likelihood model. Using Bayes' rule, the probability that an entity e is a target entity for a query q can be written as:

$$p(e|q) = \frac{p(q|e)p(e)}{p(q)}. \tag{5}$$

The probability of the query $p(q)$ is the same for all entities and can be ignored. Furthermore, we assume that the prior probability of an entity being a target entity is uniform. Thus, $p(e)$ can also be ignored. The task is to rank an entity e in response to a query q by estimating the query generation probability $p(q|e)$.

To compute $p(q|e)$, recall that our two-step query generation process assumes that users generate queries by first selecting facts and then choosing query words for each fact. Based on the query generation process, we first decompose the query q into m subqueries q_i (discussed in Section 3.2). Then the probability $p(q|e)$ can be computed as:

$$\begin{aligned} p(q|e) &= \prod_{i=1}^{m} p(q_i|e) && (6) \\ &= \prod_{i=1}^{m} \sum_{k=1}^{n} p(q_i|f_k)p(f_k|e) && (7) \\ &\simeq \prod_{i=1}^{m} \max_k p(q_i|f_k). && (8) \end{aligned}$$

Equation (6) decomposes the query into subqueries, assuming that all the subqueries are independent. Equation (7) iterates through all the factoid nodes f_k in the factoid hierarchy of an entity e. Equation (8) simplifies the computation by assuming that the underlying factoid generating subquery q_i is the factoid f_k with the highest query generation probability.

To compute $p(q_i|f_k)$, the probability of the factoid f_k generating the subquery q_i, we use the multinomial unigram language model:

$$p(q_i|f_k) = e(f_k) \prod_j p(t_i^j|f_k), \tag{9}$$

where t_i^j is the term j in the subquery q_i. $e(f_k)$ is the penalty term for factoids containing many children:

$$e(f_k) = w \cdot \frac{1}{c(f_k)}, \tag{10}$$

where $c(f_k)$ is the number of child nodes for f_k. To understand why we add this penalty term, consider a query "who is born in 2008". Suppose "Barack Obama" is described by two sentences: "born in 1961" and "elected president in 2008". When computing $p(q_i|f_k)$ for the root node, although it contains both the terms "born" and "2008", it should be penalized since the terms come from two different child nodes.

3.2 Query analysis

As mentioned earlier, we decompose the original query q into multiple factoid subqueries q_i. For long queries issued in a verbose sentence, such as "which presidents were born in 1945", dependency parsing is performed (Klein and Manning, 2003) and the resulting dependency tree is used to split the original query. For short queries issued in keywords, such as "vietnam war movies", we decompose it based on possible key concepts expressed in the query. Usually a short query only contains a single entity, which is used to segment the original query into subqueries.

Furthermore, stop structures in verbose queries is removed, following the method proposed in (Huston and Croft, 2010). Here a stop structure is defined as a phrase which provides no information regarding the information needs, such as "tell me the". We also inject target entity type information by replacing the leading "who " as "person", and "where" as "place" for all the queries.

3.3 Retrieval Process

For the purpose of retrieval, each node in the entity factoid hierarchy is regarded as a pseudo-document describing one or more factoids about the entity, and is indexed as a bag-of-words document during the preprocessing. The retrieval is performed in a two-step process. First, for each individual subquery, we retrieve top 1000 candidate entities by performing retrieval on all root nodes. This gives us an initial pool of candidate entities by merging the returned entities for subqueries. After that, for each candidate entity, we traverse its factoid hierarchy and compute the query generation probability $p(q|e)$ using Equations (8) and (9). Top ranked entities are returned as retrieval results.

4 Experiments

4.1 Dataset

We perform entity retrieval experiments using the DBpedia-Entity dataset used in (Balog and Neumayer, 2013). The dataset is a mixture of multiple entity retrieval datasets, covering entity queries of various styles such as keyword queries like "vietnam war movies" and verbose queries like "What is the capital of Canada". Some query statistics are shown in Table 2.

Query set	#query	avg($\|q\|$)	avg(#rel)
INEX-XER	55	5.5	29.7
TREC Entity	17	6.7	12.9
SemSearch ES	130	2.7	8.6
SemSearch LS	43	5.4	12.5
QALD-2	140	7.9	41.2
INEX-LD	100	4.8	36.8
Total	485	5.3	26.7

Table 2: DBpedia-Entity dataset statistics

The data corpus we use are DBpedia 3.9 and the corresponding English Wikipedia data dump on April 4, 2013. It should be noted that the original DBpedia-Entity benchmark only uses DBpedia for entity modeling (Balog and Neumayer, 2013). In our experiments, we also conducted another set of experiments which include full-text Wikipedia articles as additional entity descriptions, to evaluate the capacity of different models on handling free texts as information sources.

4.2 Comparison models and variants of our model

For comparison, we have implemented the following two existing models:

- **BM25**. BM25 is a popular document retrieval method and also used to perform entity retrieval (Balog and Neumayer, 2013). All the descriptions about an entity are aggregated into an entity pseudo-document. We use $k_1 = 1.2, b = 0.8$ for the model parameter, similar to the original papers.

- **MLM-tc**. The Mixture of Language Model represents an entity as a document with multiple fields, where each field is given a different weight for generating the query terms. MLM is often adopted to do entity retrieval (Neumayer et al., 2012). Here we adopt the MLM-tc model used in (Balog and Neumayer, 2013), where two fields are considered: title and content fields (described in Section 4.3). The parameters used are 0.8 for the title field and 0.2 for the content field.

Note that both MLM-tc and BM25 are also compared in (Balog and Neumayer, 2013), and have shown the best MAP performances among all the compared models.

For our models, the following two variants are implemented and compared.

- **Factoid Retrieval Model with Hierarchy (FRMwH)**. Our full model uses entity factoid graph as entity representation. Each factoid node is indexed as a bag-of-words document. The retrieval model described in Section 3 is employed.

- **Factoid Retrieval Model (FRM)**. This model does not use entity factoid hierarchy as entity representation. Instead, K-Means clustering algorithm is used to cluster the sentences into text clusters. Each text cluster is then indexed as a document. Compared to the FRMwH model, an entity only has a flat cluster of factoid descriptions. The same retrieval model is used.

All the four models use the same query preprocessing techniques.

4.3 Setup

The entity descriptions come from texts in Wikipedia articles and structured information from DBpedia. For DBpedia information, we consider top 1000 most frequent predicates as fields. We convert RDF predicates to free text by breaking the camelcase predicate name to terms, for example "birthPlace" is converted to "birth place". For Wikipedia texts, we first remove all markup text such as images, categories. Infoboxes are also

Model	INEX-XER		TREC Entity		SemSearch ES		SemSearch LS		QALD-2		INEX-LD		Total	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
Experiments with only DBpedia information														
BM25	.1890	.2706	.1257	.1571	.2732	.2426	.2050	.2286	.2211	.1976	.1104	.2158	.1806	.1901
MLM-tc	.1439	.2176	.1138	.1143	.2962	.2641	.1755	.1976	.1789	.1598	.1093	.2144	.1720	.1792
FRM	.2186	.2186	.1548	.1548	.2430	.2430	.2088	.2088	.2462	.2462	.1178	.1178	.1854	.1965
FRMwH	.2260	.2260	.1742	.1742	.2270	.2270	.1642	.1642	.2286	.2286	.1358	.1358	.1905	.2004
Experiments with both DBpedia and Wikipedia information														
BM25	.1313	.1887	.1374	.1667	.2916	.2526	.1867	.1833	.1552	.1253	.1698	.2680	.1848	.1821
MLM-tc	.0777	.0981	.0942	.0875	.2794	.2398	.1071	.1071	.1024	.0771	.1501	.2370	.1515	.1452
FRM	.1922	.1922	.1601	.1601	.2279	.2279	.1729	.1729	.1965	.1965	.1793	.1793	.1934	.1998
FRMwH	.2634	.2634	.1770	.1770	.2267	.2267	.1910	.1910	.2491	.2491	.1554	.1554	.2092	.2130

Table 1: Retrieval performance for various models

removed since the information is already well captured in DBpedia. Each Wikipedia article is then segmented to a list of sentences, which are considered as factoid descriptions regarding the entity.

For the BM25 model, all the descriptions about an entity are aggregated into an entity pseudo-document. For the MLMtc model, the *title* field is constructed by combining DBpedia properties whose property names are ending with "title", "name" or "label", such as "fullName" (Neumayer et al., 2012), and the *content* field is the same as the entity pseudo-document used in the BM25 model.

The inference algorithm for the entity factoid hierarchy is implemented based on the factorie package (McCallum et al., 2009). The parameters used in the inference are manually tuned on a small set of entities. The retrieval algorithms, including BM25 and Language Modeling, are implemented based on Apache Lucene[1]. For language models, Bayesian smoothing with Dirichlet priors is used, with parameter $\mu = 2000$. For FRM, to cluster the entity descriptions, we use the K-Means clustering algorithm implemented in Carrot2[2].

4.4 Results

We report two standard retrieval measures: mean average precision (MAP) and precision at 10 (P@10). Top 100 ranked entities are evaluated for each query. Two set of experiments are conducted: experiments with only DBpedia information; experiments with both DBpedia and Wikipedia information. The experiment result is shown in Table 1. To conduct the statistical significance analysis, we use two-tailed paired t-test at the 0.05 level. The symbols underline and wave underline are used to indicate significant improvement of our model compared with the BM25 and MLM-tc models respectively.

The first set of rows in Table 1 show the performance of four models using only DBpedia information. Both of our models have better overall performance. On datasets with verbose queries, such as INEX-XER and TREC Entity, both our models outperform the baseline models. One reason is that our retrieval model relies on the assumption that verbose queries can be decomposed into multiple subqueries. The second set of rows show the performance of four models using both DBpedia and Wikipedia information. After adding the additional information from Wikipedia articles, MLM-tc attains much worse performance, while BM25 performs roughly the same. One possible reason is that Wikipedia articles contain much irrelevant information regarding entities, and these two existing models cannot easily make use of additional information. In contrast, with Wikipedia full-text available, both of our proposed models achieve obviously better performances.

Our full model, FRMwH, has shown consistently better overall performance compared with the FRM model. It demonstrates that it is worthwhile to employ our proposed entity hierarchical structure for entity representation.

4.5 Analysis

For the retrieval performance, we also perform a topic-level analysis between our model FRMwH and the baseline model BM25, shown in Figure 3. The X-axis represents individual query topics, ordered by average precision difference (shown on the Y-axis). Positive Y value indicates that FRMwH performs better than the BM25 model for the query. From the figure, most of

[1]Apache Lucene: http://lucene.apache.org/
[2]Carrot[2]: http://www.carrot2.org/

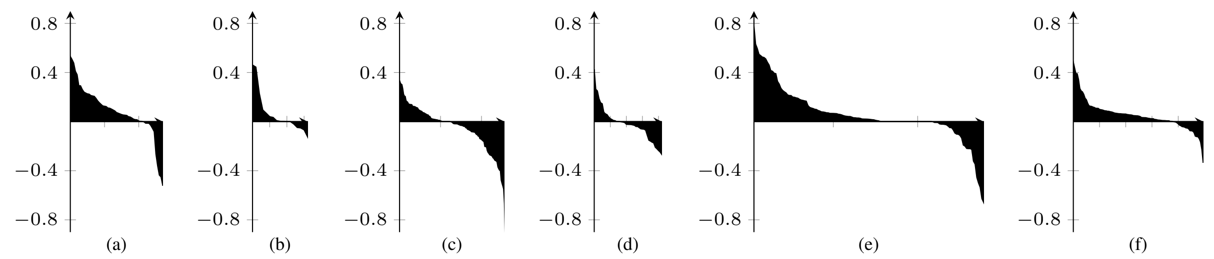

Figure 3: Topic-level differences between FRMwH and BM25. Positive values mean FRMwH is better. (a) INEX-XER; (b) TREC Entity; (c) SemSearch ES; (d) SemSearch LS; (e) QALD-2; (f) INEX-LD.

queries are affected by using FRMwH model. On the datasets with verbose queries, such as INEX-XER and TREC Entity, we can see most of the query are improved. FRMwH performs slightly worse for datasets like SemSearch ES which is mostly composed of keyword queries. For the queries that show little or no performance differences, manual inspection shows that both models fail to find any relevant results, due to the lack of supporting descriptions in Wikipedia and DBpedia.

5 Related Work

Besides the entity retrieval models reviewed in Section 1, there are models that do not maintain an explicit entity representation. Instead, they compute the entity relevance score based on the co-occurance between entities and query terms in the documents directly. Most of these models are originally proposed for expertise retrieval, where the appearance of a person name indicates the association with the expertise mentioned in the same document. Typical models include voting model (Macdonald and Ounis, 2006), graph model (Serdyukov et al., 2008), etc. However, it is not easy to generalize these models for open domain entity retrieval.

Entity models are also used in other fields besides entity retrieval. For example, entity topic models are used to perform entity prediction, classification of entity pairs, construction of entity-entity network (Newman et al., 2006), as well as entity linking (Han and Sun, 2012). These models are not suitable for our retrieval framework.

The decomposing of entity queries into factoid queries is related to query segmentation. Query segmentation has been used by search engines to support inverse lookup of words and phrases (Risvik et al., 2003; Bergsma and Wang, 2007). Our use of query decomposition is quite different compared to query segmentation. Besides query segmentation, query decomposition has also been used to facilitate the acquisition and optimization of high-order contextual term associations (Song et al., 2012).

Our work is also related to the information extraction and knowledge representation field since our framework involves extraction and aggregation of knowledge from free texts. However, most existing approaches takes two extreme ways: either extract relations based on pre-defined ontology, such as DBpedia (Lehmann et al., 2014); or cluster relation without referring to some ontology, such as OpenIE (Etzioni et al., 2011). Though our main goal is not on constructing a complete knowledge base, we do leverage both existing knowledge bases as well as free text data.

Semantic search also targets on returning answers directly (Pound et al., 2010; Blanco et al., 2011; Tonon et al., 2012; Kahng and Lee, 2012). However, they are mainly based on structured linked data, as well as structured query language like SPARQL. While this is an effective approach if we have a powerful thorough knowledge base, in practice many facts cannot be effectively represented as linked data. Only a small set of relations (thousands in DBpedia) have been defined in the ontology, such as “birthPlace”. Furthermore, even if we can define a formal representation of human knowledge, retrieve them effectively is still a problem due to the difficulty of transforming the human query into a structured query on a knowledge base.

6 Conclusions

We propose that an entity query is generated in a two-step process: users first select the facts that can distinguish target entities from the others; then choose words to express those facts. Following this motivation, we propose a retrieval framework by decomposing the original query into factoid queries. We also propose to construct an entity factoid hierarchy as the entity model for the purpose of entity retrieval. Our entity factoid hierarchy can integrate information of different granularities from both free text and structured data. Extensive experiments demonstrate the effectiveness of our framework.

References

Krisztian Balog and Robert Neumayer. 2013. A test collection for entity search in DBpedia. In *Proceedings of the 36th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 737–740.

K. Balog, P. Serdyukov, and A. P. de Vries. 2012. Overview of the TREC 2011 entity track. In *Proceedings of the Twentieth Text REtrieval Conference*.

Shane Bergsma and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *Proc. EMNLP-CoNLL*, pages 819–826.

Roi Blanco, Harry Halpin, Daniel M. Herzig, Peter Mika, Jeffrey Pound, and Henry S. Thompson. 2011. Entity search evaluation over structured web data. In *Proceedings of the 1st International Workshop on Entity-Oriented Search*, EOS '11.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 3–10.

Xianpei Han and Le Sun. 2012. An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 105–115.

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference*, pages 49–56.

Samuel Huston and W. Bruce Croft. 2010. Evaluating verbose query processing techniques. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 291–298.

Minsuk Kahng and Sang-goo Lee. 2012. Exploiting paths for entity search in rdf graphs. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 1027–1028.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2014. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.

Chunliang Lu, Lidong Bing, and Wai Lam. 2013. Structured positional entity language model for enterprise entity retrieval. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 129–138.

Craig Macdonald and Iadh Ounis. 2006. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 387–396.

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, pages 1249–1257.

Robert Neumayer, Krisztian Balog, and Kjetil Nrvg. 2012. When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In *Advances in Information Retrieval*, pages 540–543.

David Newman, Chaitanya Chemudugunta, and Padhraic Smyth. 2006. Statistical entity-topic models. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 680–686.

D. Petkova and W.B. Croft. 2006. Hierarchical language models for expert finding in enterprise corpora. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 599–608, Nov.

Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 771–780.

Hema Raghavan, James Allan, and Andrew Mccallum. 2004. An exploration of entity models, collective classification and relation description. In *Proceedings of KDD Workshop on Link Analysis and Group Detection*, pages 1–10.

K. M. Risvik, T. Mikolajewski, and P. Boros. 2003. Query segmentation for web search. In *Proceedings of the Twelfth International World Wide Web Conference (Poster session)*.

Pavel Serdyukov, Henning Rode, and Djoerd Hiemstra. 2008. Modeling multi-step relevance propagation for expert finding. In *Proceeding of the 17th ACM Conference on Information and Knowledge Mining*, pages 1133–1142.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 793–803.

Dawei Song, Qiang Huang, Peter Bruza, and Raymond Lau. 2012. An aspect query language model based on query decomposition and high-order contextual term associations. *Comput. Intell.*, 28(1):1–23, February.

Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. 2012. Combining inverted indices and structured search for ad-hoc object retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 125–134.

Qiuyue Wang and Jinglin Kang. 2012. Integrated retrieval over structured and unstructured data. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *CLEF (Online Working Notes/Labs/Workshop)*, pages 42–44.

Zhanyi Wang, Wenlong Lv, Heng Li, Wenyuan Zhou, Li Zhang, Xiao Mo, Liaoming Zhou, Weiran Xu, Guang Chen, and Jun Guo. 2011. PRIS at TREC 2011 entity track: Related entity finding and entity list completion. In *TREC*.

Qiuyue Wang, Jaap Kamps, Georgina Ramírez Camps, Maarten Marx, Anne Schuth, Martin Theobald, Sairam Gurajada, and Arunav Mishra. 2012. Overview of the inex 2012 linked data track. In *CLEF (Online Working Notes/Labs/Workshop)*.

Michael Wick, Sameer Singh, and Andrew McCallum. 2012. A discriminative hierarchical model for fast coreference at large scale. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 379–388.

Michael Wick, Sameer Singh, Harshal Pandya, and Andrew McCallum. 2013. A joint model for discovering and linking entities. In *CIKM 2013 Workshop on Automated Knowledge Base Construction*, pages 67–72.