Exploiting Entity Linking in Queries for Entity Retrieval

Faegheh Hasibi
Norwegian University of Science and Technology
faegheh.hasibi@idi.ntnu.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

Svein Erik Bratsberg
Norwegian University of Science and Technology
sveinbra@idi.ntnu.no

ABSTRACT

The premise of entity retrieval is to better answer search queries by returning specific entities instead of documents. Many queries mention particular entities; recognizing and linking them to the corresponding entry in a knowledge base is known as the task of entity linking in queries. In this paper we make a first attempt at bringing together these two, i.e., leveraging entity annotations of queries in the entity retrieval model. We introduce a new probabilistic component and show how it can be applied on top of any term-based entity retrieval model that can be emulated in the Markov Random Field framework, including language models, sequential dependence models, as well as their fielded variations. Using a standard entity retrieval test collection, we show that our extension brings consistent improvements over all baseline methods, including the current state-of-the-art. We further show that our extension is robust against parameter settings.

CCS Concepts

•**Information systems** → **Retrieval models and ranking;**

Keywords

Entity retrieval; entity linking; semistructured retrieval

1. INTRODUCTION

The past decade has witnessed an emergence of entity-oriented information access technology [30]. Among these are two main tasks that have been extensively addressed: (i) answering information needs with specific entities, a problem referred to as *entity retrieval* [7, 25, 35, 36, 38, 46], and (ii) identifying and disambiguating entities in text, a process known as *entity linking* [8, 19, 22]. Both these tasks represent key building blocks for semantic search [30] and are typically backed by a large-scale knowledge base. Despite this common ground, the two tasks have so far been studied mostly on their own, as standalone problems. We say mostly, as entity ranking, to a limited extent, has already met entity linking: most entity linking approaches involve an entity retrieval component for collecting candidate entities for a given text segment, see, e.g., [8, 19, 22]. The other direction, utilizing entity linking for entity retrieval, to the best of our knowledge, has not been explored yet. This study is a first attempt at bridging this gap by performing entity linking on search queries and using the resulting annotations to improve entity retrieval.

It has been shown in prior work that entity retrieval can be improved by leveraging semantic annotations of the query, such as target entity types or related entities, see, e.g., [4, 10, 24, 39]. These studies, using the TREC Entity [3] and INEX-XER [17] benchmarking platforms, assume that semantic annotations are provided as part of the definition of the information need, i.e., complement the keyword query. Further, these test suites comprise a homogeneous set of queries, where all queries are of the same general type or even follow some predefined template (e.g., input entity, target type, and required relation for the related entity finding task at TREC [3]). In this work, we use a heterogeneous query set, ranging from short keyword queries to natural language questions, and obtain entity annotations automatically. It is further worth pointing out that virtually all existing work is limited to term-based representation of entities. A few studies stand as exceptions, but those address the entity ranking task in some specific flavor, such as list completion [11], or focus on a particular property of entities, like types [4]. Our approach, on the other hand, considers both term- and entity-based representations of entities for general-purpose entity retrieval; see Figure 1.

It is worth relating our efforts to the large body of prior work that has shown that leveraging information about entity annotations of queries can improve document retrieval performance [9, 16, 27, 28, 43, 44]. Importantly, this task is very different from ours: we search for entities in a (manually curated) knowledge base where entities are first-class citizens. This stands in contrast with document retrieval, where the entity annotations are a result of some automated process, which always involves a degree of uncertainty. Not only the task, but the techniques used for utilizing entity annotations are also different; in document retrieval entities are typically used for query expansion or as simple features in a learning-to-rank framework (see §2.3). We, on the other hand, represent and match entities directly as a separate component in the retrieval model.

Against this background, the main research question driving our work is this: *What is a theoretically sound way of extending term-based entity retrieval models with the capability of leveraging linked entities in the query?* To address this question, we introduce (i) a new component for matching entity annotations of the query with entity relationships recorded in a knowledge base and (ii) a general framework for leveraging this component into the term-based models. Our framework is based on the Markov Random Field (MRF) model [31]. There are several reasons for this particular choice of framework, including its solid theoretical foundations, good empirical performance, the fact that it can encompass a number of

ICTIR '16, September 12–16, 2016, Newark, Delaware, USA.

DOI: http://dx.doi.org/10.1145/2970398.2970406

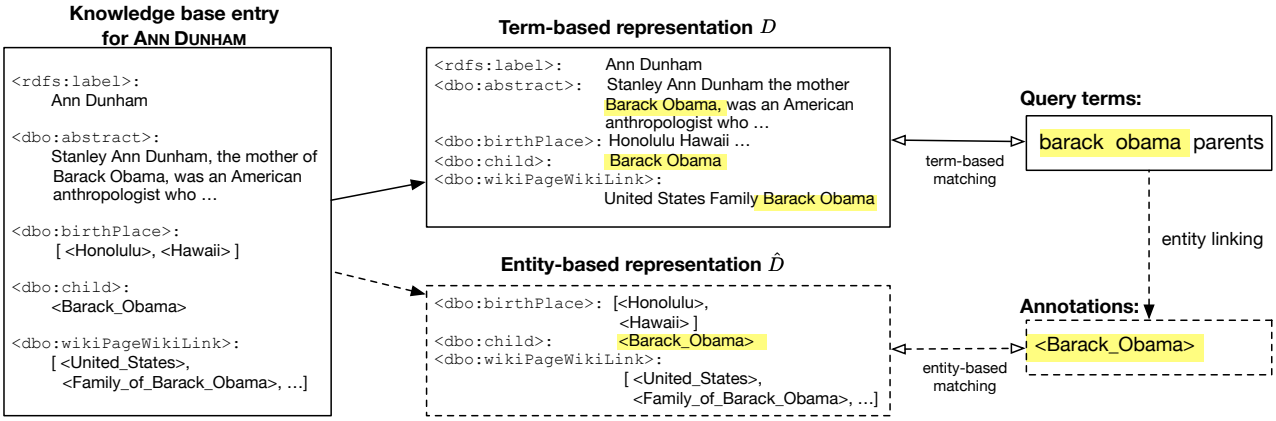


Figure 1: Demonstration of term- and entity-based representation of entities. The query terms match against the term-based representation of entity ANN DUNHAM, while the entity annotations of the queries match against the entity-based representation. The dashed parts indicate the novel elements of our work.

existing retrieval models, and last but not least, the current state-of-the-art in ad-hoc entity retrieval is also based on MRF [46]. Within this framework, we introduce a new component for matching the linked entities from the query. This component, termed ELR (for "Entity Linking incorporated Retrieval"), may be seen as an extension that can be applied on top of any text-based retrieval model that can be instantiated in the MRF framework. Entity matches are facilitated by an additional entity-based representation that preserves entity relationships as recorded in the knowledge base; see the block denoted with $\hat{D}$ in Figure 1. We address a number of technical and modeling issues that stem from the differences between terms and entities, including the spareness of entity-based representations compared to term-based ones, varying number of entity annotations per query, dealing with uncertainties involved with entity linking, and mapping of entities to fields. We conduct experiments on a test collection consisting of close to 500 heterogeneous queries and show that our model can consistently and significantly improve upon standard language models [45], semi-structured [25], and term-dependence models [31, 46], and outperforms the current state-of-the-art on ad-hoc entity retrieval by over 6% in terms of MAP. In addition, we demonstrate its robustness against parameter setting and entity linking configuration.

The resources developed within this paper are made publicly available at http://bit.ly/ictir2016-elr.

2. RELATED WORK

Our work falls in the intersection of entity retrieval, entity linking, and exploiting query annotation in retrieval.

2.1 Entity retrieval

One main theme in entity retrieval research concerns the representation of entities; once a term-based representation is created, entities can be ranked using traditional retrieval models, much like documents. Early work, especially in the context of expert search, obtains such representations by considering mentions of the given entity across the document collection [2, 4]. The INEX 2007-2009 Entity Retrieval track (INEX-XER) [17, 18] studies entity retrieval in Wikipedia, while the INEX 2012 Linked Data track goes one step further and considers Wikipedia articles together with RDF properties from the DBpedia and YAGO2 knowledge bases [42]. Much of the recent work represents entities as fielded documents, extracted from a knowledge base [1, 7, 46] or from multiple information sources [20]. In this work, we introduce a new representation layer, referred to as entity-based representation, based on entity relationships stored in the knowledge base.

Entity retrieval models can be categorized into two main groups: semistructured retrieval models [1, 36, 46] and learning-to-rank approaches [20, 41] . Our focus in this paper is on the first category, where fielded representations of entities are ranked using fielded variations of standard document retrieval models, e.g., BM25F [40] or the mixture of language models [37]. This is indeed the predominant approach for ad-hoc entity retrieval [1, 7, 36]. In a recent effort, Zhiltsov et al. [46] extend the Sequential Dependence Model [31] to multi-field representation of entities. Within this context, the choice of fields and estimation of field weights remain a challenge. Our work also addresses these challenges by using a principled and parameter-free estimation method [25].

Entity retrieval has also been explored in the context of specific tasks, such as list completion at INEX-XER [17, 18, 39] or related entity finding at the TREC Entity track [3, 5]. In both these cases, types or entities are provided as part of the topic definition. Our approach obtains the entity annotations for queries automatically.

2.2 Entity linking

Early work on entity linking has focused on long texts, based on contextual and semantic similarities between a document and candidate entities [13, 15, 26, 33, 34]. More recently, the focus has slightly shifted towards annotating short texts such as tweets and queries [8, 19, 21, 29]. Entity linking for short texts is challenging, mainly because of the lack of context. In case of queries, there are additional efficiency considerations, as entity linking needs to be performed on-the-fly. TAGME is one of the early systems that addresses entity linking in short texts and has received due attention since [23]. It is one of the best performing systems, both in terms of efficiency and effectiveness [12, 14], and also offers a public API.

The problem of entity linking in queries has recently been identified as a separate task, different from conventional entity linking for short/long texts [12, 22]. The difference lies in the number of entities that can be assigned to a mention: for short/long texts a single entity is ought to be returned for each mention, while entity linking in queries can associate multiple entities with a single mention, if the ambiguity cannot be resolved. Hasibi et al. [22] discuss that entity linking in queries should be presented as a set of entity linking interpretations, where each interpretation consists of semantically related entities. For our approach, we are interested in the annotation of individual entities and are not concerned with

entity linking interpretations. Therefore, any type of entity linking systems can be used with our approach. Due to reproducibility considerations [23], we employ the TAGME API as a black-box entity linker and incorporate the resulting annotations into the entity retrieval system.

2.3 Exploiting query annotations in retrieval

Exploiting semantic annotations of queries for various retrieval tasks has attracted due attention over the recent years. Examples include the INEX and TREC entity benchmarking efforts as already discussed in §2.1. Much of the recent work has focused on incorporating entity-related information to improve document retrieval [9, 16, 27, 28, 43, 44]. There are three major differences between this line of work and ours. First and foremost, the task is different; the referred approaches address ad-hoc document retrieval, while we search for entities in a knowledge base. Second, the way entity annotations are utilized is different; we aim to directly incorporate entity annotations into the retrieval model, while existing approaches either employ query expansion techniques [9, 16, 27, 44] or operate in a latent entity space [28, 43]. Finally, our approach requires only the identifiers of the linked entities, thereby making it a generic and effective model (also in terms of implementation), while others rely on additional entity-related information from the knowledge base.

Schuhmacher et al. [41] address a variant of the ad-hoc entity ranking task for informational queries, which is "close in spirit to ad-hoc document search" [41]. Even though this task is different from ours, there are some similarities. They also leverage entity annotations of queries, but they do so in a learning-to-rank framework by introducing two binary features: whether the query entity (i) contains or (ii) is related to the candidate entity or not. The findings on the merits of these features are somewhat inconclusive; while they are shown to be influential on the Robust04 dataset, they were less helpful on ClueWeb12 dataset. The conclusiveness of these results might be further limited by the low number of queries (25 and 22 for Robust and ClueWeb12, respectively).

3. BACKGROUND

In this section, we describe the Markov Random Field (MRF) model [31], which is the basis of our proposed approach (following in §4). We further discuss two specific variations of the MRF model: the Sequential Dependence Model [31] in §3.2 and the Fielded Sequential Dependence Model [46] in §3.3.

3.1 The Markov Random Field model

Markov Random Field models for information retrieval were first introduced by Metzler and Croft [31] to model the dependencies between query terms. Given a document D and a query Q, the goal of these models is to compute the joint probability $P(Q, D)$:

$$P(D|Q) = \frac{P(Q,D)}{P(Q)} \stackrel{rank}{=} P(Q,D) \tag{1}$$

This probability is estimated based on a Markov Random Field, which is a common practice to compute the joint probabilities of random variables. For document retrieval, a MRF is defined by a graph G with nodes consisting of query terms q_i and the document D, and edges representing the dependence between the nodes. The joint probability over variables of the graph G is computed as:

$$P_\Lambda(Q,D) = \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \psi(c;\Lambda),$$

where $C(G)$ is the set of cliques in G and $\psi(c;\Lambda) = \exp[\lambda_c f(c)]$ is a non-negative *potential function*, parametrized by the weight λ_c and the *feature function* $f(c)$. The parameter Z_Λ serves as a normalization factor, which is generally ignored due to computational infeasibility. Substituting all these elements into Eq. (1), the final ranking function becomes:

$$P(D|Q) \stackrel{rank}{=} \sum_{c \in C(G)} \lambda_c f(c). \tag{2}$$

This ranking function provides a solid theoretical basis for a wide spectrum of retrieval models: from traditional unigram-based models to more sophisticated ones involving n-grams as well as additional task- or domain-specific features [6, 39]. To build a ranking function, all one needs to do is to define the graph structure and the potential functions over the graph cliques.

3.2 Sequential Dependence Model

The Sequential Dependence Model (SDM) is a popular MRF-based retrieval model, which provides a good balance between retrieval effectiveness and efficiency [31]. In the underlying graph of this model, only adjacent query terms are connected to each other, meaning that query terms are sequentially dependent on each other; i.e., the white nodes in Figure 2. Under this assumption, the potential functions are defined for two types of cliques: (i) 2-cliques involving a query term and the document, (ii) cliques containing two contiguous terms and the document. The potential function for the first type of cliques is:

$$\psi(q_i, D; \Lambda) = \exp[\lambda_T f_T(q_i, D)], \tag{3}$$

where $f_T(q_i, D)$ is the feature function for the query term q_i and the document D. There are two possibilities for the second type of cliques (two terms): either the terms occur contiguously in the query or they do not. These two cases make up the potential functions for *ordered* and *unordered* matches, and are denoted by the O and U subscripts, respectively:

$$\begin{aligned}\psi(q_i, q_{i+1}, D; \Lambda) = &\exp[\lambda_O f_O(q_i, q_{i+1}, D) + \\ &\lambda_U f_U(q_i, q_{i+1}, D)].\end{aligned} \tag{4}$$

By substituting the two potential functions $\psi(q_i, D; \Lambda)$ in Eq. (3) and $\psi(q_i, q_{i+1}, D; \Lambda)$ in Eq. (4) into Eq. (2), and factoring out the λ parameters, the SDM ranking function becomes:

$$\begin{aligned}P(D|Q) \stackrel{rank}{=} &\lambda_T \sum_{q_i \in Q} f_T(q_i, D) + \\ &\lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \\ &\lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D),\end{aligned} \tag{5}$$

where the parameters should meet the constraint of $\lambda_T + \lambda_O + \lambda_U = 1$. The specific feature functions are set as follows:

$$f_T(q_i, D) = \log[\frac{tf_{q_i,D} + \mu \frac{cf_{q_i}}{|C|}}{|D| + \mu}] \tag{6}$$

$$f_O(q_i, q_{i+1}, D) = \log[\frac{tf_{\#1(q_i,q_{i+1}),D} + \mu \frac{cf_{\#1(q_i,q_{i+1})}}{|C|}}{|D| + \mu}] \tag{7}$$

$$f_U(q_i, q_{i+1}, D) = \log[\frac{tf_{\#uwN(q_i,q_{i+1}),D} + \mu \frac{cf_{\#uwN(q_i,q_{i+1})}}{|C|}}{|D| + \mu}], \tag{8}$$

where tf_D is the frequency of the term(s) in the document D and cf denotes the total number of occurrences of the term(s) in the entire collection. The function $\#1(q_i, q_{i+1})$ searches for the exact

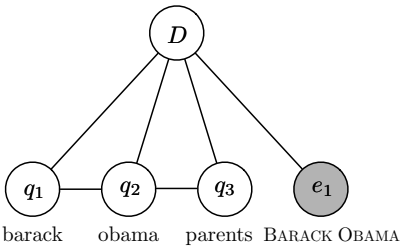


Figure 2: Graphical representation of the ELR model for the query "barack obama parents". Here, all the terms are sequentially dependent and the phase "barack obama" is linked to the entity BARACK OBAMA.

match of the phrase q_i, q_{i+1}, while $\#uwN(q_i, q_{i+1})$ counts the co-occurrence of terms within a window of N words (where N is set to 8 based on [31]). The parameter μ is the Dirichlet prior, which is taken to be the average document length in the collection.

Putting all these together, SDM is basically the weighted sum of language model scores obtained from three sources: (i) query terms, (ii) exact match of query bigrams, and (iii) unordered match of query bigrams. Our approach in §4 employs the same sequential dependence assumption that SDM does.

3.3 Fielded Sequential Dependence Model

The Fielded Sequential Dependence Model (FSDM) [46] extends the SDM model to support structured document retrieval. In essence, FSDM replaces the document language model of feature functions (Eqs. (6), (7), and (8)) with those of the Mixture of Language Models (MLM) [37]. Given a fielded representation of a document (e.g., title, body, anchors, metadata, etc. in the context of web document retrieval), MLM computes a language model probability for each field and then takes a linear combination of these field-level models. Hence, FSDM assumes that a separate language model is built for each document field and then computes the feature functions based on fields $f \in \mathcal{F}$, with $\mathcal{F}$ being the universe of fields. For individual terms, the feature function becomes:

$$f_T(q_i, D) = \log \sum_f w_f^T \frac{tf_{q_i,D_f} + \mu_f \frac{cf_{q_i,f}}{|C_f|}}{|D_f| + \mu_f}, \quad (9)$$

while ordered and unordered bigrams are estimated as:

$$f_O(q_i, q_{i+1}, D) = \log \sum_f w_f^O \frac{tf_{\#1(q_i,q_{i+1}),D_f} + \mu_f \frac{cf_{\#1(q_i,q_{i+1}),f}}{|C_f|}}{|D_f| + \mu_f} \quad (10)$$

$$f_U(q_i, q_{i+1}, D) = \log \sum_f w_f^U \frac{tf_{\#uwN(q_i,q_{i+1}),D_f} + \mu_f \frac{cf_{\#uwN(q_i,q_{i+1}),f}}{|C_f|}}{|D_f| + \mu_f}. \quad (11)$$

The parameters w_f are the weights for each field, which are set to be non-negative with the constraint $\sum_f w_f = 1$. Zhiltsov et al. [46] trained both the field weights (w_f) and the feature function weights ($\lambda_T, \lambda_O, \lambda_U$ in Eq. (5)) in two stages using the Coordinate Ascent algorithm [32].

4. THE ELR APPROACH

This section presents our approach for incorporating entity linking into entity retrieval. We start by introducing our general MRF-based framework in §4.1, and continue with describing the feature functions in §4.2 and fielded representation of entities in §4.3.

4.1 Model

Our *Entity Linking incorporated Retrieval* (ELR) approach is an extension of the MRF framework for incorporating entity annotations into the retrieval model. We note, without detailed elaboration, that ELR is applicable to a wide range of retrieval problems where documents, or document-based representations of objects, are to be ranked, and entity annotations are available to be leveraged in the matching of documents and queries. Our main focus in this paper, however, is limited to entity retrieval; entity annotations are an integral part of the representation here, cf. Figure 1. (This is unlike to traditional document retrieval, where documents would need to be annotated by an automated process that is prone to errors). To show the generic nature of our approach, and also for the sake of notational consistency with the previous section, we shall refer to documents throughout this section. We detail how these documents are constructed for our particular task, entity retrieval, in §4.3.

Our interest in this work lies in incorporating entity annotations and not in creating them. Therefore, entity annotations of the query are assumed to have been generated by an external entity linking process, which we treat much like a black box. Formally, given an input query $Q = q_1 ... q_n$, the set of linked entities is denoted by $E(Q) = \{e_1, ..., e_m\}$. We do not impose any restrictions on these annotations, i.e., they may be overlapping and a given query span might be linked to multiple entities. It might also be that $E(Q)$ is an empty set. Further, we assume that annotations have confidence scores associated with them. For each entity $e \in E(Q)$, let $s(e)$ denote the confidence score of e, with the constraint of $\sum_{e \in E(Q)} s(e) = 1$.

The graph underlying our model consists of document, term, and entity nodes. As shown in Figure 2, we assume that the query terms are sequentially dependent on each other, while the annotated entities are independent of each other and of the query terms. Based on this assumption, the potential functions are computed for three types of cliques: (i) 2-cliques consisting of edges between the document and a term node, (ii) 3-cliques consisting of the document and two term nodes, and (iii) 2-cliques consisting of edges between the document and an entity node. The potential functions for the first two types are identical to the SDM model (Eqs. (3) and 4). We define the potential function for the third clique type as:

$$\psi_E(e, D; \Lambda) = \exp[\lambda_E f_E(e, D)],$$

where λ_E is a free parameter and $f_E(e, D)$ is the feature function for the entity e and document D (to be defined in §4.2). By substituting all feature functions into Eq. (2), the MRF ranking function becomes:

$$P(D|Q) \overset{rank}{=} \sum_{q_i \in Q} \lambda_T f_T(q_i, D) + \sum_{q_i, q_{i+1} \in Q} \lambda_O f_O(q_i, q_{i+1}, D) + \sum_{q_i, q_{i+1} \in Q} \lambda_U f_U(q_i, q_{i+1}, D) + \sum_{e \in E(Q)} \lambda_E f_E(e, D).$$

This model introduces an additional parameter for weighing the importance of entity annotations, λ_E, on top of the three parameters ($\lambda_{\{T,O,U\}}$) from the SDM model (cf. §3.2). There is a crucial difference between entity-based and term-based matches with regards

to the λ parameters. The number of cliques for term-based matches is proportional to the length of the query ($|Q|$ for unigrams and $|Q|-1$ for ordered and unordered bigrams), which makes them compatible (directly comparable) with each other, irrespective of the length of the query. Therefore, in SDM, $\lambda_{\{T,O,U\}}$ are taken out of the summations (cf. Eq. (5)) and can be trained without having to worry about query length normalization. The parameter λ_E, however, cannot be treated the same manner for two reasons. Firstly, the number of annotated entities for each query varies and it is independent of the length of the query. For example, a long natural language query might be annotated with a single entity, while shorter queries are often linked to several entities, due to their ambiguity. Secondly, we need to deal with varying levels of uncertainty that is involved with entity annotations of the query. The confidence scores associated with the annotations, which are generated by the entity linking process, should be integrated into the retrieval model.

To address the above issues, we re-write the λ parameters as a parameterized function over each clique and define them as:

$$\lambda_T(q_i) = \lambda_T \frac{1}{|Q|},$$
$$\lambda_O(q_i, q_{i+1}) = \lambda_O \frac{1}{|Q|-1},$$
$$\lambda_U(q_i, q_{i+1}) = \lambda_U \frac{1}{|Q|-1},$$
$$\lambda_E(e) = \lambda_E s(e),$$

where $|Q|$ is the query length and $s(e)$ is the confidence score of entity e obtained from the entity linking step. Considering this parametric form of the λ parameters, our final ranking function takes the following form:

$$\begin{aligned} P(D|Q) \overset{rank}{=} \; & \lambda_T \sum_{q_i \in Q} \frac{1}{|Q|} f_T(q_i, D) + \\ & \lambda_O \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_O(q_i, q_{i+1}, D) + \\ & \lambda_U \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_U(q_i, q_{i+1}, D) + \\ & \lambda_E \sum_{e \in E(Q)} s(e) f_E(e, D), \end{aligned} \tag{12}$$

where the free parameters λ are placed under the constraint of $\lambda_T + \lambda_O + \lambda_U + \lambda_E = 1$. This model ensures that the scores for the different type of matches (i.e., term, ordered window, unordered window, and entities) are normalized and the λ parameters, which are to be trained, are not influenced by the length of the query or by the number of linked entities. In addition, it provides us with a general ranking framework that can encompass various retrieval models. If the λ_O and λ_U parameters are set to zero, the model is an extension of unigram based models, such as LM and MLM. Otherwise, it extends SDM and FSDM. We also note that due to the normalizations applied to the different set of matches, the *full dependence* variant of MRF model [31] could also be instantiated in our framework; this, however, is outside the scope of this study.

4.2 Feature functions

Feature functions form an essential part of MRF-based models. We now discuss the estimation of these for the ELR model. For all feature functions, we use a fielded document representation of entities, as it is a common and effective approach for entity retrieval, see, e.g., [1, 7, 20, 35, 46]. The first three feature functions in Eq. (12), f_T, f_O, and f_U, are computed as defined in Eqs. (9), (10), and (11), respectively.

Let us then turn to defining the function $f_E(e, D)$ in Eq. (12), which is a novel feature introduced by our ELR model. This function measures the goodness of the match between an entity e linked in the query and a document D. These matches are facilitated by an entity-based representation of documents. For each document D an entity-based representation $\hat{D}$ is obtained by ignoring document terms and considering only entities. In the context of our work, the entity represented by document D stands in typed relationships with a number of other entities, as specified in the knowledge base. The various relationships are modeled as fields in the document. Consider the example in Figure 1, where the document represents the entity ANN DUNHAM, who is being linked to the entity BARACK OBAMA (via the relationship <dbo:child>). This entity-based representation differs from the traditional term-based representation in at least two important ways. Firstly, each entity appears at most once in each document field. Secondly, if an entity appears in a field, then it should be considered a match, irrespective of what other entities may appear in that field. Consider again the example in Figure 1, where the field <dbo:birthplace> has multiple values, HONOLULU and HAWAII. Then, if either of these entities is linked in the query, that should account for a perfect match against this particular field, irrespective of how many other locations are present in that field. Motivated by these observations, we define the feature function f_E as:

$$f_E(e, D) = \log \sum_{f \in \mathcal{F}} w_f^E \Big[(1-\alpha) tf_{\{0,1\}(e,\hat{D}_f)} + \alpha \frac{df_{e,f}}{df_f} \Big], \tag{13}$$

where the linear interpolation implements the Jelinek-Mercer smoothing method, with α set to 0.1, and $tf_{\{0,1\}(e,\hat{D}_f)}$ indicates whether the entity e is present in the document field $\hat{D}_f$ or not. For the background model, we employ the notion of document frequency as follows: $df_{e,f} = |\{\hat{D} | e \in \hat{D}_f\}|$ is the total number of documents that contain the entity e in field f and $df(f) = |\{\hat{D} | \hat{D}_f \neq \emptyset\}|$ is the number of documents with a non-empty field f.

All of the feature functions f_T, f_O, f_U, and f_E involve free parameters w_f, which control the field weights. Zhiltsov et al. [46] set these type of parameters using a learning algorithm, which leads to a large number of parameters to be trained (the number of feature functions times the number of fields). Instead, we employ a parameter-free estimation of field weights, using field mapping probabilities, introduced in the Probabilistic Retrieval Model for Semistructured Data (PRMS) [25]. This probability infers the importance of each field, with respect to a given query term, based on collection statistics of that term. Specifically, the probability of a field f, from the universe of fields $\mathcal{F}$, is computed with respect to a given term t as follows:

$$P(f|t) = \frac{P(t|f)P(f)}{\sum_{f' \in \mathcal{F}} P(t|f')P(f')}. \tag{14}$$

Here, $P(f)$ is the prior probability of field f, which is set proportional to the frequency of the field (across all documents in the collection), and $P(t|f)$ is estimated by dividing the number of occurrences of term t in field f by the sum of term counts in f across the whole collection. We compute the probability $P(f|t)$ for all query terms, ordered and unordered bigrams, and use the resulting values for the weights w_f^T, w_f^O, and w_f^U (used in Eqs. (9)-(11)), respectively. The weights w_f^E (used in Eq. (13)) are also estimated using Eq. (14), but this time we compute this probability for entities instead of terms (i.e., t is replaced with e).

Employing the mapping probability $P(f|.)$ instead of free parameters w_f [46] has three advantages. First, the field mapping probability specifies field importance for each query term (or bigram) individually, while the w_f parameters are the same for all

"finland"		FINLAND	
Field name	Mapping prob.	Field name	Mapping prob.
<dcterms:subject>	0.210	<dbo:country>	0.223
<dbo:wikiPageWikiLink>	0.178	<dbo:wikiPageWikiLink>	0.201
types	0.168	contents	0.189
contents	0.113	<dbo:birthPlace>	0.170
<rdfs:comment>	0.089	<dbo:hometown>	0.053
<dbo:abstract>	0.070	<dbo:location>	0.047
<rdfs:label>	0.069	<dbo:nationality>	0.041
names	0.059	<dbo:deathPlace>	0.034
<foaf:isPrimaryTopicOf>	0.040	<dbo:locationCountry>	0.028
yago:<rdf:type>	0.001	<dbo:ground>	0.0129

Table 1: Selected fields with the corresponding mapping probabilities for the term "finland" and the entity FINLAND.

query terms (or bigrams). Second, the number of free parameters in the feature functions f_T, f_O, f_U, and f_E reduces from $4 * |\mathcal{F}|$ to zero. Hence, the final model is more robust and can be employed in various settings, without risking overfitting. Lastly and most importantly, estimating the field weights this way allows us to have a query-specific selection of fields, depending on the linked entity, as opposed to having pre-trained (fixed) field weights.

4.3 Fielded representation of entities

We now detail how the term-based and entity-based representation are obtained for entities, from the knowledge base entry (i.e., subject-predicate-object triples) describing the entity. One of the challenges of working with a fielded document-based representation of entities is the appropriate selection of fields. While grouping SPO triples by predicates and mapping each predicate to a separate document field is straightforward, retrieval can become highly inefficient because of the large number of fields [35]. Previous work has suggested a number of solutions to alleviate this problem by reducing the number of fields, which can be summarized under two main categories: (i) selecting a subset of fields that are considered and (ii) grouping fields together into a handful of predefined categories. When using the first approach, predicates are commonly ordered by frequency and a rank-based cutoff is applied, e.g., top 1000 in [1]. There are two choices for assigning the field weights in this setting: to simply use uniform values for all fields or to employ some estimation technique (such as the field mapping probabilities in the PRMS model) as training is generally infeasible due to the large number of fields. Examples of the second technique, referred to as "predicate folding" in [35], include grouping fields into a handful of predetermined categories based on type [35, 46] or manually determined importance [7]. It has been shown in [36] that it is possible to achieve solid performance even with as few as two fields, "title" and "content." One main advantage of predicate folding is that the estimation of field weights becomes tractable. The disadvantage is that a large part of the semantics associated with the individual predicates is discarded.

In this work we combine these two strategies to get the best of both approaches. We employ predicate folding for three designated fields: names, types, and content (see §5.1). In addition, we consider all fields, which are not included in names or types, on their own. From this combined set, we then select the top-N most frequent fields across the whole knowledge base and use them for the term-based entity representation.

The entity-based representation requires a different field selection procedure from the above, as entities (SPO triples with an URI value as object) occur less often and follow an entirely different pattern than terms. For instance, the entity FINLAND mostly occurs in the <dbo:country> and <dbo: birthPlace> fields, while the entity ANCIENT ROMAN ARCHITECTURE often appears in the <dbo:architecturalStyle> field. This illustrates that it is not desirable to have the same (and fixed) set of fields for all entities, but field selection should be performed on an entity-specific manner. Therefore, for each entity, we select the top-N fields, from the entity-based representations, that the entity occurs in. As this computation can be performed offline, it does not negatively impact on the efficiency of retrieval. Table 1 shows an excerpt of the mapping probability distribution for a given term and entity.

5. EXPERIMENTAL SETUP

This section presents our experimental setup, including the data set (§5.1), field selection (§5.2), parameter settings (§5.3), and how entity linking is performed (§5.4).

5.1 Data

We use DBpedia version 3.9 as our knowledge base along with the DBpedia-entity test collection [1].

Indices. For our experiments, we created two fielded indices from subject-predicate-object triples: a *term-based index*, where all entities (URI objects) are resolved to terms, and an *entity-based index*, where only URI objects are kept. The former index is used to compute the unigram and bigram term probabilities (Eqs. (9)-(11)), while the latter is employed for the entity probability computations (Eq. (13)). We built the indices using Lucene and made use of SpanNearQuery to get the statistics for ordered and unordered phrases. Our indices are confined to entities having a name and a short abstract (i.e. fields <rdfs:label> and <rdfs:comment>), resulting in a total of 3,984,580 entities. They contain the top-1000 most frequent DBpedia predicates as fields, together with three other fields: (i) the names field, which is the constitution of entity predicates <rdfs:label>, <foaf:name>, and redirected entities; (ii) the types field, which contains <rdf:type> and attribute names ending in "subject"; (iii) the contents field, which holds the contents of all entity fields except entity links in other languages (<owl:sameAs>). In the term-based index, terms are lowercased and stopped using the default Lucene stopwords list, and all URIs are replaced with the name of the corresponding entity. In the entity-based index, only URIs are indexed and all literal objects are ignored. In addition, the URI of each entity itself is also added to the contents field in the entity-based index.

Queries. We evaluate the effectiveness of our models using the DBpedia-entity collection [1], which comprises 485 queries from a number of entity retrieval benchmarking campaigns. Following [6], queries are stopped using a handful of stop patterns ("which", "who",

Query subset	#queries	Avg. len	#rel
SemSearch ES	130	2.7	1115
ListSearch	115	5.6	2390
INEX_LD	100	4.8	3680
QALD-2	140	7.9	5773
Total	485	5.3	12958

Table 2: Query subsets of the DBpedia-entity test collection.

"what", "where", "give me", "show me") to improve entity linking and initial retrieval performance. We perform stopwords removal after the entity linking step, using the default Lucene stopwords list. We break down retrieval results according to the four categories suggested by Zhiltsov et al. [46]:

- **SemSearch ES**: Keyword queries targeting specific entities, which are often ambiguous (e.g., "madrid," "hugh downs").
- **ListSearch**: Combination of INEX-XER, SemSearch LS, and TREC Entity queries, targeting a list of entities that match a certain criteria (e.g., "Airports in Germany," "the first 13 american states").
- **INEX_LD**: General keyword queries, involving a mixture of names, types, relations, and attributes (e.g., "Eiffel," "vietnam war movie," "gallo roman architecture in paris").
- **QALD-2**: Natural language queries (e.g., "which country does the creator of miffy come from," "give me all female russian astronauts").

Table 2 provides descriptive statistics on these query subsets.

5.2 Field selection

The number of fields used in the term-based representation is a parameter shared by all but two of the evaluated models (single-field LM and SDM). We examined retrieval performance against a varying number of fields ($n = 10^i, i = 0, 1, 2, 3$), see Figure 3. Note that fields are ordered by frequency. It is clear from the figure that the best results are obtained when the top 10 most frequent fields are used. We used this setting in all our experiments, unless stated otherwise. For consistency, we also used the same setting, i.e., top 10 fields, for the entity-based representation.

5.3 Parameter setting

This section describes the parameter settings used in our experiments. The Dirichlet prior μ in language models is set to the average document/field length across the collection. The unordered window size N in Eqs. (8) and (11) is chosen to be 8, as suggested in [31, 46]. To estimate the λ parameters involved in SDM, FSDM, and our approach, we employ the Coordinate Ascent (CA) algorithm [32] and directly optimize Mean Average Precision (MAP). CA is a commonly used optimization technique, which iteratively optimizes a single parameter while holding all other parameters fixed. We make use of the CA implementation provided in the RankLib framework and set the number of random restarts to 3. Following [46], we estimate the λ parameters of SDM, FSDM*, and ELR-based approaches using 5-fold cross validation for each of the 4 query subsets separately. We note that Zhiltsov et al. [46] train both the λ and w parameters (Eq. (5)-(11)) for the FSDM model. As we use different entity representation from [46] (with 10 as opposed to 5 fields), training parameters in this manner would result in cross-validation of 33 parameters for each query subset, which would be prone to overfitting. We avoid this issue by employing the PRMS field mapping probability for field weights w

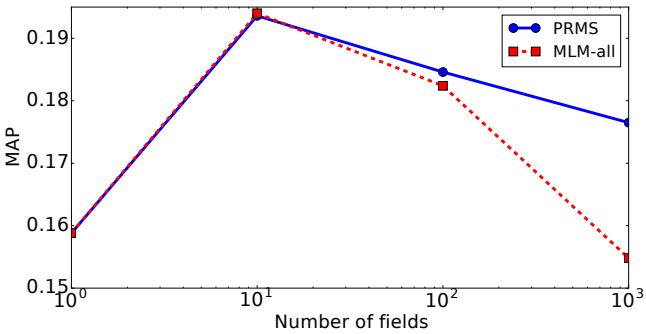


Figure 3: Effect of varying number of fields on MAP.

(i.e., Eq. (14)). Therefore, our implementation of FSDM slightly deviates from the original paper [46]; in acknowledgement of this distinction, we will refer to our implementation as FSDM*.

For all experiments, we employ a two stage retrieval method: first an initial set of top 1000 results is retrieved using Lucene's default search settings, then this set is re-ranked with the specific retrieval model (using an in-house implementation). Evaluation scores are reported on the top 100 results. To perform cross-validation, we randomly create train and test folds from the initial result set, and use the same folds throughout all the experiments. To measure statistical significance we employ a two-tailed paired t-test and denote differences at the 0.01 and 0.05 levels using the ▲ and △ symbols, respectively.

5.4 Entity linking

Entity linking is a key component of the ELR approach. For the purpose of reproducibility, all the entity annotations in this work are obtained using an open source entity linker, TAGME [19], accessed through its RESTful API.[1] TAGME is one of the best performing entity linkers for short queries [12, 14]. As suggested in the API documentation, we use the default threshold 0.1 in our experiments; we analyze the effect of the threshold parameter in §6.5.

6. RESULTS AND ANALYSIS

We begin by enumerating our research questions, then present a series of experiments conducted to answer them.

6.1 Research questions

We address the following research questions:

- **RQ1**: Can entity retrieval performance be improved by incorporating entity annotations of the query? (§6.2)
- **RQ2**: How are the different query subsets impacted by ELR? (§6.3)
- **RQ3**: How robust is our method with respect to parameter settings? (§6.4)
- **RQ4**: What is the impact of the entity linking component on end-to-end performance? (§6.5)

6.2 Overall performance

To find out whether entity linking in queries can improve entity retrieval performance (RQ1), we compare a number of entity retrieval approaches proposed in the literature. The first four (LM,

[1] http://tagme.di.unipi.it/

Model	SemSearch ES		ListSearch		INEX-LD		QALD-2	
	MAP	**P@10**	**MAP**	**P@10**	**MAP**	**P@10**	**MAP**	**P@10**
LM	.2485	.2008	.1529	.1939	.1129	.2210	.1132	.0729
LM + ELR	.2531$^{\triangle}$ (+1.8%)	.2008	.1688$^{\blacktriangle}$ (+10.4%)	.2096	.1244$^{\blacktriangle}$ (+10.2%)	.2330	.1241$^{\blacktriangle}$ (+9.6%)	.0836$^{\blacktriangle}$
PRMS	.3517	.2685	.1722	.2270	.1184	.2240	.1180	.0893
PRMS + ELR	.3573 (+1.6%)	.2700	.1956$^{\blacktriangle}$ (+14.1%)	.2417	.1303$^{\blacktriangle}$ (+10%)	.2330	.1343$^{\blacktriangle}$ (+13.8%)	.1064$^{\blacktriangle}$
SDM	.2669	.2108	.1553	.1948	.1167	.2250	.1369	.0750
SDM + ELR	.2641 (-1%)	.2115	.1689$^{\triangle}$ (+8.8%)	.2174$^{\blacktriangle}$	.1264$^{\blacktriangle}$ (+8.3%)	.2340	.1472$^{\triangle}$ (+7.5%)	.0857
FSDM*	.3563	.2692	.1777	.2165	.1261	.2290	.1364	.0921
FSDM* + ELR	.3581 (+.5%)	.2677	.1973$^{\triangle}$ (+11%)	.2391$^{\blacktriangle}$	.1332 (+5.6%)	.2330	.1583$^{\triangle}$ (+16%)	.1086$^{\blacktriangle}$

Table 4: Results of ELR approach on different query types. Significance is tested against the line above; the numbers in parentheses show the relative improvements, in terms of MAP.

Model	MAP		P@10	
LM	0.1588		0.1664	
MLM-tc	0.1821		0.1786	
MLM-all	0.1940		0.1965	
PRMS	0.1936		**0.1977**	
SDM	0.1719		0.1707	
FSDM*	**0.2030**		0.1973	
LM + ELR	0.1693$^{\blacktriangle}$	(+6.6%)	0.1757$^{\blacktriangle}$	(+5.6%)
MLM-tc + ELR	0.1937$^{\blacktriangle}$	(+6.4%)	0.1895$^{\blacktriangle}$	(+6.1%)
MLM-all + ELR	0.2082$^{\blacktriangle}$	(+7.3%)	0.2054$^{\triangle}$	(+4.5%)
PRMS + ELR	0.2078$^{\blacktriangle}$	(+7.3%)	0.2085$^{\blacktriangle}$	(+5.5%)
SDM + ELR	0.1794$^{\blacktriangle}$	(+4.4%)	0.1812$^{\blacktriangle}$	(+6.1%)
FSDM* + ELR	**0.2159**$^{\blacktriangle}$	(+6.3%)	**0.2078**$^{\blacktriangle}$	(+5.3%)

Table 3: Retrieval results for baseline models (top) and with ELR applied on top of them (bottom). Significance is tested against the corresponding baseline model. Best scores are in boldface.

MLM-tc, MLM-all, and PRMS) are language modeling-based methods that were introduced as standard baselines for the DBpedia-entity test collection [1]. The other two (SDM and FSDM) are taken from [46], the work that reported the best results on this collection so far. Specifically, the baseline models considered are:

- **LM**: The standard language modeling approach [45], against the `contents` field.
- **MLM-tc**: The Mixture of Language Models [37] with two fields, `names` and `contents`, with weights 0.2 and 0.8, respectively, as suggested in [36].
- **MLM-all**: The Mixture of Language Models using the top 10 fields with equal weights.
- **PRMS**: The Probabilistic Retrieval Model for Semistructured Data [25], using the top 10 fields.
- **SDM**: The Sequential Dependence Model [31], against the `contents` field.
- **FSDM***: The Fielded Sequential Dependence Model [46] on the top 10 fields, with field weights estimated using PRMS.

The top section of Table 3 displays the results of these baselines, all implemented from scratch.[2] The bottom part of Table 3 shows the results we get by applying ELR on top of these baselines. We observe consistent improvements over all baselines; the relative ranking of models remains the same (LM < SDM < MLM-tc < MLM-all < PRMS < FSDM*), but their performance is improved by 4.4–7.3% in terms of MAP, and by 4.5–6.1% in terms of P@10. All improvements are statistically significant. Based on these results, we answer our first research question positively: entity annotations of the query can indeed improve entity retrieval performance.

For the analysis that follows later in this section we retain four of these models: LM, PRMS, SDM, and FSDM*. This selection enables us to make a meaningful and consistent comparison across two dimensions: (i) single vs. multiple fields (LM and SDM vs. PRMS and FSDM*), and (ii) term independence vs. dependence (LM and PRMS vs. SDM and FSDM*).

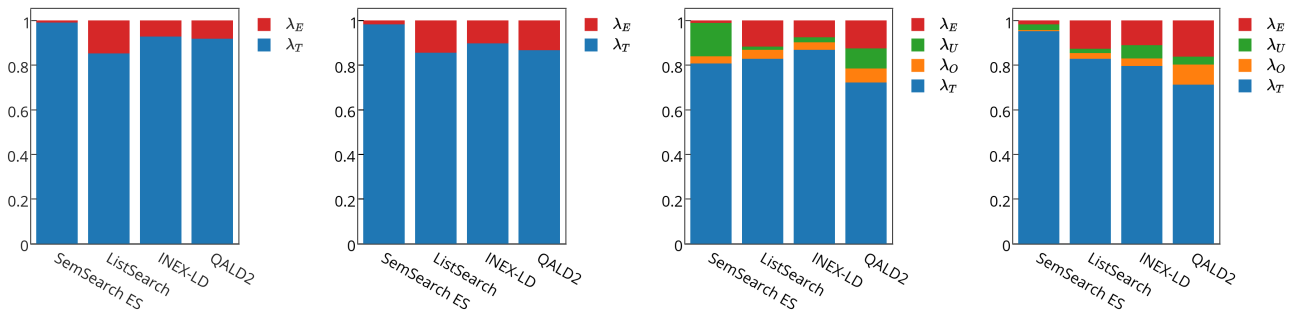
6.3 Breakdown by query subsets

How are the different query subsets impacted by ELR (RQ2)? Intuitively, we expect ELR to improve the effectiveness of queries that mention entities plus contain some additional terms (e.g., specifying an attribute or relation). Queries that mention a single entity, without any modifiers, are less likely to benefit from our approach.

Table 4 provides a breakdown of results by query type. We find that the biggest improvements are obtained for the ListSearch and QALD-2 queries (+7.5–16% in terms of MAP and +6.5–19.1% in terms of P@10). The queries in these sets often seek entities related to other entities; a lot can be gained here from entity linking. The INEX-LD queries are also significantly improved by ELR (with the exception for FSDM*), but the relative improvements are smaller than for ListSearch and QALD-2 (here, it is +5.6–10.2% for MAP and +1.7–5.4% for P@10), but still significant in all but one case. This set is more diverse than the other two and comprises a mixture of short entity queries, type queries, and long natural language style queries. Finally, on the SemSearch ES subset, ELR could make a significant difference only for the weakest baseline, LM. These are short keyword queries, which are already handled effectively by a fielded representation (cf. PRMS and FSDM; also note that adding term dependence does not improve performance).

To understand how much importance is attributed to entity-based matches, we plot the values of the $\lambda_{\{T,O,U,E\}}$ parameters for each

[2]We note the slight differences compared to the numbers reported in [1] and [46]. One major source of the differences is that we use a different DBpedia version, v3.9 with the updated DBpedia-entity test collection, as opposed to v3.7 (used both in [1] and [46]). Our MLM-all and PRMS results are better than [1] because we use the top 10 fields, while top 1000 fields are used in [1] (cf. Figure 3). Compared to [46], we got higher scores for PRMS and lower ones for FSDM. The main reason behind this is the different choice of fields. Furthermore, as explained in §5.3, field weights are trained for each query subset in [46], while we employ a parameter-free estimation of field weights based on PRMS.



(a) LM+ELR (b) PRMS+ELR (c) SDM+ELR (d) FSDM*+ELR

Figure 4: Values of the λ parameters (λ_T: unigrams, λ_O: ordered bigrams, λ_U: unordered bigrams, λ_E: entities) in our experiments, by the different query subsets (trained using Coordinate Ascent).

query subset in Figure 4. The values are obtained by averaging the trained parameter values across all folds of the cross-validation process. We can observe a similar trend across all retrieval methods: ListSearch and QALD-2 queries are assigned the highest λ_E values, INEX-LD gets a somewhat lower but still sizable portion of the distribution, while for SemSearch ES it bears little importance.

Based on Table 4 and Figure 4, we conclude that different query types are impacted differently by the ELR method. The results confirm our hypothesis that ELR can improve complex (ListSearch and QALD-2) as well as heterogeneous (INEX-LD) query sets, which involve entity relationships. On the other hand, short keyword queries, referring to a single, albeit often ambiguous, entity (SemSearch ES) are mostly unaffected.

6.4 Parameter settings

How sensitive is ELR to the choice of λ parameters (RQ3)? To answer this question, we compare two configurations: (i) default parameter settings, and (ii) parameters trained using the CA algorithm. The default parameters are set as follows. For SDM and FSDM, we follow [31, 32] and set $\lambda_T = 0.8$, $\lambda_O = 0.1$, and $\lambda_U = 0.1$. For the other models with ELR applied, we set λ_E to the single best performing value across the entire query set; that is, we do not train it separately for the different query subsets, like before. The resulting configurations are: (i) $\lambda_T = 0.9$ and $\lambda_E = 0.1$ for LM + ELR and PRMS + ELR, and (ii) $\lambda_T = 0.8$, $\lambda_O = 0.05$, $\lambda_U = 0.05$, and $\lambda_E = 0.1$ for SDM + ELR and FSDM* + ELR.

Table 5 compares retrieval results using default and trained parameters. Note that LM and PRMS do not involve any parameters, hence the empty cells. We find that the results are robust, i.e., ELR can improve the performance of term-based models, even with default parameter values. MAP differences are significant for all methods, except SDM + ELR. (For that model, the default λ_E value is higher than it would be optimal for SemSearch ES queries, thereby reducing overall retrieval effectiveness.) This experiment also confirms that our improvements are not a result of overfitting.

6.5 Impact of entity linking

What is the impact of the entity linking component on end-to-end entity retrieval performance (RQ4)? Entity linking systems typically involve a threshold parameter that defines the required degree of certainty for linking entities. This threshold for TAGME ranges between 0 and 1, where 0 returns the maximum number of entities and 1 returns no entity. To answer the above research question, we measure retrieval performance while varying the entity linking threshold value. Figure 5 reports the results for the best performing model, FSDM* + ELR, for both trained and default λ parameters.

Model	Default params.		Trained params.	
	MAP	**P@10**	**MAP**	**P@10**
LM	0.1588	0.1664		
LM + ELR	0.1668$^{\triangle}$	0.1724	0.1693$^{\blacktriangle}$	0.1757$^{\blacktriangle}$
PRMS	0.1936	0.1977		
PRMS + ELR	0.2028$^{\blacktriangle}$	0.2035	0.2078$^{\blacktriangle}$	0.2085$^{\blacktriangle}$
SDM	0.1672	0.1685	0.1719	0.1707
SDM + ELR	0.1721	0.1722	0.1794$^{\blacktriangle}$	0.1812$^{\blacktriangle}$
FSDM	0.1969	0.1973	0.2030	0.1973
FSDM + ELR	0.2043$^{\blacktriangle}$	0.1996	0.2159$^{\blacktriangle}$	0.2078$^{\blacktriangle}$

Table 5: Comparison of default vs. trained λ parameters over all queries. Significance is tested against the line above.

Apart from the small fluctuations in the 0.4–0.6 range, retrieval performance is shown to improve as the entity linking threshold is lowered. This observation implies that ELR is robust with respect to entity linking; considering more entity annotations, even those with low confidence, improves retrieval performance. The entity linker currently used allows for the annotation of overlapping entity mentions, but it returns a single entity for each mention. In future work it might be worth experimenting with multiple entities per mention, especially in highly ambiguous situations, as our framework seems to be able to benefit from having more annotations.

7. CONCLUSION

This paper represents a first attempt at incorporating entity linking into entity retrieval. We have presented a novel retrieval approach that complements term-based retrieval models with entity-based matches, using automatic means to annotate queries with entities. Our model is based on Random Markov Fields and is presented as a general framework, in which the entity-based matching component can be applied to a wide range of entity retrieval models, including standard language models, term dependence models, and their fielded variations. We have applied our approach as an extension to various state-of-the-art entity retrieval models and have shown significant and consistent improvements over all of them. The results have also shown that our model especially benefits complex and heterogeneous queries (natural language, type and relation queries), which are considered difficult queries in the context of entity retrieval. We have further demonstrated the robustness of our approach against parameter setting and entity linker configuration.

There are several directions for future work. First is to explore the effectiveness of our model using the entity linking systems specif-

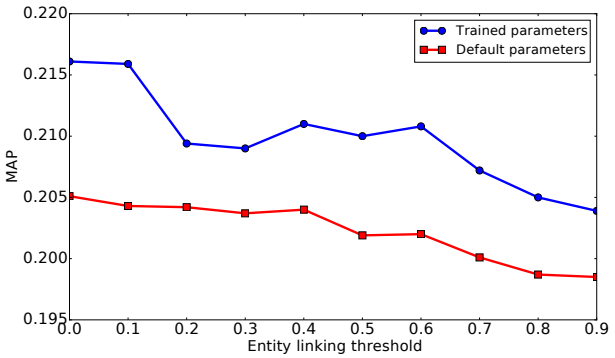


Figure 5: Effect of changing entity linking threshold (TAGME) on the performance of FSDM* + ELR model.

ically designed for queries, where a mention may be linked to multiple entities [12, 22]. Additional avenues for future work include applying our model to other retrieval problems and considering different flavors of semantic annotations, e.g., entity types.

Acknowledgments. We thank Hamed Zamani for his helpful comments during the preparation of the final version of this paper.

References

[1] K. Balog and R. Neumayer. A test collection for entity search in DBpedia. In *Proc. of SIGIR*, pages 737–740, 2013.
[2] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *Proc. of SIGIR*, pages 43–50, 2006.
[3] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. Overview of the TREC 2009 entity track. In *Proc. of TREC*, 2010.
[4] K. Balog, M. Bron, and M. De Rijke. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29:22:1–22:31, 2011.
[5] K. Balog, P. Serdyukov, and A. P. de Vries. Overview of the TREC 2011 entity track. In *Proc. of TREC*, 2012.
[6] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *Proc. of WSDM*, pages 31–40, 2010.
[7] R. Blanco, P. Mika, and S. Vigna. Effective and efficient entity search in RDF data. In *Proc. of ISWC*, pages 83–97, 2011.
[8] R. Blanco, G. Ottaviano, and E. Meij. Fast and space-efficient entity linking in queries. In *Proc. of WSDM*, pages 179–188, 2015.
[9] W. C. Brandão, R. L. T. Santos, N. Ziviani, E. S. de Moura, and A. S. da Silva. Learning to expand queries using entities. *JASIST*, 65(9): 1870–1883, 2014.
[10] M. Bron, K. Balog, and M. de Rijke. Ranking related entities: Components and analyses. In *Proc. of CIKM*, pages 1079–1088, 2010.
[11] M. Bron, K. Balog, and M. de Rijke. Example based entity search in the web of data. In *Proc. of ECIR*, pages 392–403, Berlin, Heidelberg, 2013.
[12] D. Carmel, M.-W. Chang, E. Gabrilovich, B.-j. P. P. Hsu, and K. Wang. ERD' 14: Entity recognition and disambiguation challenge. *SIGIR Forum*, 48:63–77, 2014.
[13] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. Learning relatedness measures for entity linking. In *Proc. of CIKM*, pages 139–148, 2013.
[14] M. Cornolti, P. Ferragina, and M. Ciaramita. A framework for benchmarking entity-annotation systems. In *Proc. of WWW*, pages 249–260, 2013.
[15] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL*, pages 708–716, 2007.
[16] J. Dalton, L. Dietz, and J. Allan. Entity query feature expansion using knowledge base links. In *Proc. of SIGIR*, pages 365–374, 2014.
[17] A. de Vries, A.-M. Vercoustre, J. Thom, N. Craswell, and M. Lalmas. Overview of the INEX 2007 entity ranking track. *Focused Access to XML Documents*, 4862:245–251, 2008.
[18] G. Demartini, T. Iofciu, and A. de Vries. Overview of the INEX 2009 entity ranking track. *Focused Retrieval and Evaluation*, 6203: 254–264, 2010.
[19] P. Ferragina and U. Scaiella. TAGME: On-the-fly annotation of short text fragments (by Wikipedia entities). In *Proc. of CIKM*, pages 1625–1628, 2010.
[20] D. Graus, M. Tsagkias, W. Weerkamp, E. Meij, and M. de Rijke. Dynamic collective entity representations for entity ranking. In *Proc. of WSDM*, 2016.
[21] S. Guo, M.-W. Chang, and E. Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *Proc. of HLT-NAACL*, pages 1020–1030, 2013.
[22] F. Hasibi, K. Balog, and S. E. Bratsberg. Entity linking in queries: Tasks and evaluation. In *Proc. of ICTIR*, pages 171–180, 2015.
[23] F. Hasibi, K. Balog, and S. E. Bratsberg. On the reproducibility of the TAGME entity linking system. In *Proc. of ECIR*, pages 436–449, 2016.
[24] R. Kaptein, P. Serdyukov, A. De Vries, and J. Kamps. Entity ranking using Wikipedia as a pivot. In *Proc. of CIKM*, pages 69–78, 2010.
[25] J. Kim, X. Xue, and W. B. Croft. A probabilistic retrieval model for semistructured data. In *Proc. of ECIR*, pages 228–239, 2009.
[26] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *Proc. of KDD*, pages 457–466, 2009.
[27] R. Li, L. Hao, X. Zhao, P. Zhang, D. Song, and Y. Hou. A query expansion approach using entity distribution based on Markov Random Fields. In *Proc. of AIRS*, pages 387–393, 2015.
[28] X. Liu and H. Fang. Latent entity space: A novel retrieval approach for entity-bearing queries. *Inf. Retr.*, 18(6):473–503, 2015.
[29] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *Proc. of WSDM*, pages 563–572, 2012.
[30] E. Meij, K. Balog, and D. Odijk. Entity linking and retrieval for semantic search. In *Proc. of WSDM*, pages 683–684, 2014.
[31] D. Metzler and W. B. Croft. A Markov Random Field model for term dependencies. In *Proc. of SIGIR*, pages 472–479, 2005.
[32] D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Inf. Retr.*, 10(3):257–274, 2007.
[33] R. Mihalcea and A. Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proc. of CIKM*, pages 233–242, 2007.
[34] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *Proc. of CIKM*, pages 509–518, 2008.
[35] R. Neumayer, K. Balog, and K. Nørvåg. On the modeling of entities for ad-hoc entity search in the Web of Data. In *Proc. of ECIR*, pages 133–145, 2012.
[36] R. Neumayer, K. Balog, and K. Nørvåg. When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In *Proc. of ECIR*, pages 540–543, 2012.
[37] P. Ogilvie and J. Callan. Combining document representations for known-item search. *Proc. of SIGIR*, pages 143–150, 2003.
[38] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the Web of Data. In *Proc. of WWW*, pages 771–780, 2010.
[39] H. Raviv, D. Carmel, and O. Kurland. A ranking framework for entity oriented search using Markov Random Fields. In *Proc. of International Workshop on Entity-Oriented and Semantic Search*, JIWES, 2012.
[40] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proc. of CIKM*, pages 42–49, 2004.
[41] M. Schuhmacher, L. Dietz, and S. Paolo Ponzetto. Ranking entities for Web queries through text and knowledge. In *Proc. of CIKM*, pages 1461–1470, 2015.
[42] Q. Wang, J. Kamps, G. R. Camps, M. Marx, A. Schuth, M. Theobald, S. Gurajada, and A. Mishra. Overview of the INEX 2012 linked data track. In *CLEF 2012 Evaluation Labs and Workshop, Online Working Notes*, 2012.
[43] C. Xiong and J. Callan. EsdRank: Connecting query and documents through external semi-structured data. In *Proc. of CIKM*, pages 951–960, 2015.
[44] C. Xiong and J. Callan. Query expansion with Freebase. In *Proc. of ICTIR*, pages 111–120, New York, NY, USA, 2015.
[45] C. Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2:137–213, 2008.
[46] N. Zhiltsov, A. Kotov, and F. Nikolaev. Fielded sequential dependence model for ad-hoc entity retrieval in the Web of Data. In *Proc. of SIGIR*, pages 253–262, 2015.