# Improving Entity Retrieval on Structured Data

Besnik Fetahu, Ujwal Gadiraju and Stefan Dietze

L3S Research Center, Leibniz Universität Hannover, Germany
{fetahu, gadiraju, dietze}@L3S.de

**Abstract.** The increasing amount of data on the Web, in particular of Linked Data, has led to a diverse landscape of datasets, which make entity retrieval a challenging task. Explicit cross-dataset links, for instance to indicate co-references or related entities can significantly improve entity retrieval. However, only a small fraction of entities are interlinked through explicit statements. In this paper, we propose a two-fold entity retrieval approach. In a first, offline preprocessing step, we cluster entities based on the *x–means* and *spectral* clustering algorithms. In the second step, we propose an optimized retrieval model which takes advantage of our precomputed clusters. For a given set of entities retrieved by the BM25F retrieval approach and a given user query, we further expand the result set with relevant entities by considering features of the queries, entities and the precomputed clusters. Finally, we re-rank the expanded result set with respect to the relevance to the query. We perform a thorough experimental evaluation on the Billions Triple Challenge (BTC12) dataset. The proposed approach shows significant improvements compared to the baseline and state of the art approaches.

## 1 Introduction

The emergence of the Web of Data, particularly supported through W3C standards such as RDF and the Linked Data principles [2], has led to a wide range of semi-structured RDF data being available on the Web. Data is spread across datasets, complemented through a growing amount of entities as part of structured annotations of Web documents, using RDFa or Microformats. Recent studies have shown that approximately 26% of pages already contain structured annotations [19].

Web data forms a highly heterogeneous knowledge-graph spanning an estimated 100 billion triples [17], with a wide variety of languages, schemas, domains and topics [7]. Even though a large number of entities and concepts are highly overlapping, that is they represent the same or related concepts, explicit links are still limited and often concentrated within large established knowledge graphs, like DBpedia [1].

The entity-centric nature of the Web of data has led to a shift towards tasks related to entity and object retrieval [3, 21] or entity-driven text summarization [6]. Major search engine providers such as Google and Yahoo! already exploit such data to facilitate semantic search using knowledge graphs, or as part of similar efforts such as the *EntityCube-Renlifang* project at Microsoft Research [14]. In such scenarios, data is aggregated from a range of sources calling for efficient means to search and retrieve entities in large data graphs. Specifically, *entity retrieval* (also known as Ad-Hoc Object retrieval) [17, 21] aims at retrieving relevant entities given a user query. The result is a ranked list of entities [3]. By simply applying standard keyword search algorithms, like the BM25F, promising results can be achieved. A common practice is to construct indexes over the textual descriptions (*literals*) of entities.

In most cases, queries are entity centric. However, there are a large number of queries that are also topic-based, e.g. '`U.S. Presidents`'. Therefore, approaches like [21] have proposed retrieval techniques that make use of the explicit links between entities in the WoD for results or query expansion. For instance, following `owl:sameAs` or `rdfs:seeAlso` predicates from `dbp:Barack_Obama`, one can retrieve co-references or highly related entities. However, considering the size of the WoD such statements are very sparse (see Figure 1a).

In this work, we propose a method for improving entity retrieval results in two aspects. We improve the task by *expanding* and *re-ranking* the result set from a baseline retrieval model (BM25F). Sparsity of explicit links is addressed through clustering of entities based on their similarity, using a combination of lexical and structural features. Consequently, we expand the result set with additional entities from the *cluster space* (clusters with which the baseline entities are associated), retrieved from the baseline.

For the expanded result set, there is a need for re-ranking. The re-ranking considers the similarity of entities to the user query, and their relevance likelihood based on the corresponding entity type, defined as *query type affinity*. We empirically model the query type affinity between the entity type in a query (e.g. *'Barack Obama'* `isA Person`) and the entity types in the result set (see Section 3.2).

In terms of *scalability* and *efficiency*, the clustering process is carried out offline, where we *bucket* entities of particular types together before clustering. This improves the efficiency by reducing the run-time of the clustering algorithms (Section 4.2 and 7.3). The entity retrieval, expansion and re-ranking on the other hand are performed online and the computational overhead is negligible (Section 5 and 7.3).

Our experimental evaluation is carried out on the BTC12 dataset [9], and using the SemSearch[1] query dataset. The individual steps in our approach are evaluated through a reliable crowdsourced evaluation approach. The results show that the proposed approach outperforms existing basslines for the entity retrieval task.

The main contributions of our work are as follows: (a) an entity retrieval model combining keyword search and entity clustering, and (b) an entity ranking model considering the query type affinity w.r.t the set of relevant entity types.

## 2 Related Work

A large portion of queries issued in Web search engines target entities or contain semantic resources (such as types, relations and attributes) [17] as a primary intent. Consequently, the identification of entity-centric queries has become of particular concern for commercial search engines serving as a means to narrow the search space and to provide contextual query results [12]. Thus, the traditional task of Ad-hoc Document Retrieval (ADR) [11] is moving towards an entity retrieval task [17]. Hence, instead of top–*k* document retrieval that match a keyword query, the task and therefore the results are increasingly becoming entity-centric.

[1] `http://km.aifb.kit.edu/ws/semsearch10/`

Following this direction, Tonon et al. [21] proposed a hybrid approach based on query expansion and relevance feedback techniques on top of the BM25 ranking function to build an entity retrieval framework. In contrast to this work, we use the state-of-the-art BM25F [5, 20] to assign varying degrees of importance to different parts of a document. Further, through an offline pre-processing step we are able to infer links between similar entities for the retrieval process. This is particularly important when considering datasets that have less links between entities, a significant feature of the work by Tonon et al [21]. Another advantage of adopting BM25F is penalising documents/entities, consisting of long textual literals, in the final ranking [10]. Sindice [15] is another approach focusing on indexing RDF documents. It supports data discovery and integration by taking advantage of DBpedia entities as a source to actively index resources. The process performed by Sindice plays a key role in centralising disparate data sources on the Web. The adoption of entities and foremost entity types (topics) is also supported by [3] in the recommendation of entities in Web search. Our approach can benefit Sindice by indexing documents following a topic-based fashion.

Zhiltsov and Agichtein [23] propose a learning to rank approach, where they model the relations between entities through a various set of features, such as language models and other query related features (e.g query length). Finally, through tensor matrix factorisation they find latent similarities between entities, later used in their learning to rank model. One major disadvantage of this approach is that it is supervised, hence, unlikely to perform reasonably well on ad-hoc entity search tasks.

## 3 Approach and Overview

In this section, we motivate and define our work in the context of the addressed challenge, and provide an overview of our approach.

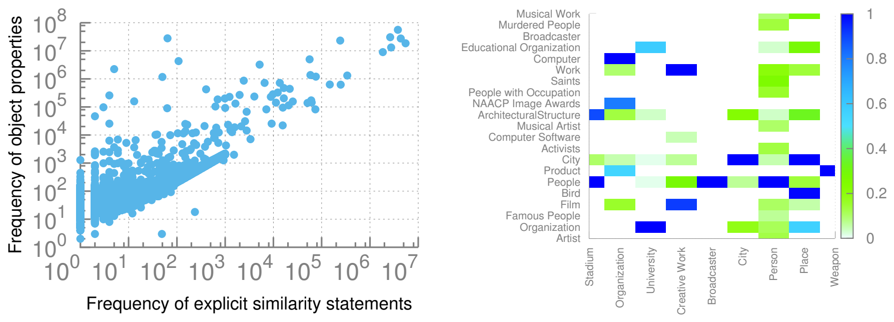
### 3.1 Preliminaries

The *entity retrieval (ER) task*, also known as ad-hoc object retrieval, is concerned with retrieving a top–$k$ ranked set of entities from dataset for a given a user query $q$. User queries are typically entity centric. A *dataset* in our case is a set of triples $\langle s, p, o \rangle$, where $s$ is the *subject* (the URI of an entity), $p$ is the *predicate*, and $o$ is the *object* (a URI or a literal). An *entity* profile of $e$ is the set of triples sharing the same subject URI $s$. The *type* of an entity is determined by the object of the triple $t_e = \langle s, \texttt{rdf:type}, o \rangle$. Additionally, we define the *query type* $t_q$, corresponding to the entity type in $q$, e.g. *'Barack Obama'*, hence $t_q$ `hasType Person`.

### 3.2 Motivation: Result Set Expansion and Query Affinity in Entity Retrieval

Recent studies [21] have shown that *explicit similarity statements*, which indicate some form of similarity or equivalence between entities, for instance through predicates such as `owl:sameAs`, are useful for improving entity retrieval results as retrieved through approaches like BM25F, i.e. improving significantly on standard precision/recall metrics. However, such explicit similarity statements usually are sparse and often focused towards a few well established datasets like DBpedia, Freebase etc. One main reason is that these datasets represent known, and well structured graphs, which show a comparably high proportion of such dedicated similarity statements, in turn linking similar entities within and beyond their original namespace.

In Figure 1a we show the total amount of explicit similarity statements (on the x–axis) that interlink entities in the BTC12 dataset. Referring to [21], here we specifically consider triples of the form $\langle e, p, e' \rangle$ where the predicate $p \in \{$`owl:sameAs`, `skos:related`, `dbp:wikiPageExternalLink`, `dbp:wikiPageDisambiguates`, `dbp:synonym`$\}$. These are plotted against the total number of *object properties* (y–axis), where each point in the plot represents a graph in the BTC12 collection. From the figure, it is obvious that the number of explicit similarity statements is very sparse, considering the size of the dataset.



(a) (b)

Fig. 1: (a) Number of explicit similarity statements in contrast to the frequency of object property statements overall, shown for all data graphs. (b) Query type affinity shows the query type and the corresponding entity types from the retrieved and relevant entities.

Nonetheless, missing links between entities can be partially remedied by computing their pair-wise similarity, thereby complementing statements like `owl:sameAs` or `skos:related`. Given the semi-structured nature of RDF data, graph-based and lexical features can be exploited for similarity computation. Particularly, lexical features derived from literals provided by predicates such as *rdfs:label* or *rdfs:description* are prevalent in LOD. Our analysis on the BTC12 dataset reveals that a large portion of entities (around 90%) have an average literal length of 50 characters.

Furthermore, while the query type usually is not considered in state of the art ER methods, we investigated its correlation with the corresponding entity types from the query result set. We refer to a ground truth[2] using the BTC10 dataset. We focus only on relevant entities for $q$. We analyze the *query type affinity* of the result sets by assessing the likelihood of an entity in the results to be of the same type as the query type. Figure 1b shows the query type affinity. On the x-axis we show the query type, whereas on the y-axis the corresponding relevant entity types are shown. Figure 1b shows that

[2] `http://km.aifb.uni-karlsruhe.de/ws/semsearch10/Files/assess`

most queries have high affinity with a specific entity type, with the difference being the query type `Person`, where relevant entities have a wider range of types.

Our work exploits such *query type affinity* to improve the ranking of entities for a query $q$ (see Section 5). Based on these observations, we argue that (a) *entity clustering* can remedy the lack of existing linking statements and (b) entity *re-ranking* considering the *query type affinity* are likely to improve the entity retrieval task.

## 3.3 Approach Overview

In this work we propose a novel approach for the *entity retrieval* task which builds on the observations described earlier. Figure 2 shows an overview of the proposed approach. The individual steps are outlined below and described in detail in Section 4 and 5. We distinguish between two main steps: (I) *offline pre-processing*, including step I.a and I.b in the following overview, and (II) *online entity retrieval*, covered by steps II.a to II.c.

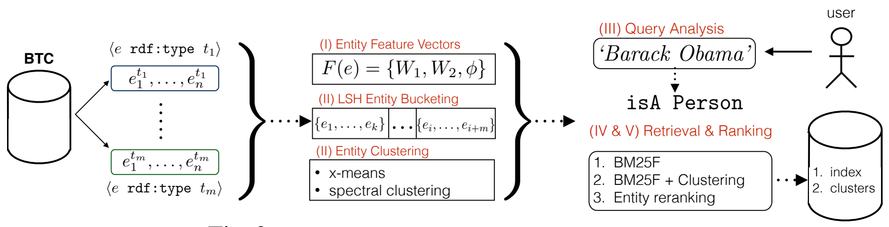

BTC
$\langle e$ rdf:type $t_1\rangle$
$e_1^{t_1}, \dots, e_n^{t_1}$
$e_1^{t_m}, \dots, e_n^{t_m}$
$\langle e$ rdf:type $t_m\rangle$
(I) Entity Feature Vectors
$F(e) = \{W_1, W_2, \phi\}$
(II) LSH Entity Bucketing
$\{e_1, \dots, e_k\}$ $\{e_i, \dots, e_{i+m}\}$
(II) Entity Clustering
• x-means
• spectral clustering
user
(III) Query Analysis
'Barack Obama'
isA Person
(IV & V) Retrieval & Ranking
1. BM25F
2. BM25F + Clustering
3. Entity reranking
1. index
2. clusters

Fig. 2: Overview of the *entity retrieval* approach.

**I.a Entity Feature Vectors:** We construct the entity feature vector as follows: $F(e) = \{\mathbf{W}_1(e), \mathbf{W}_2(e), \phi\}$, where $\mathbf{W}_1(e)$ and $\mathbf{W}_2(e)$ represent the *unigrams* and *bigrams* extracted from literals of $e$, and $\phi$ represents the structural features.

**I.b Entity Bucketing & Clustering:** is used to compute implicit relationships between entities emerging from their feature vectors. For the sake of efficiency, before we proceed with entity clustering, we exploit the locality-sensitive hashing (LSH) algorithm for bucketing.

**II.a Query Analysis:** As part of the retrieval task, we initially analyse the given user queries $q$. From the query terms, which typically represent named entities, we determine the type of the named entity, e.g. '`Location`' in order to support the query type affinity-based reranking at a later stage.

**II.b Entity Retrieval:** In the retrieval process, we rely on a combination of standard IR approaches, like BM25F and further expand the result set with entities showing a high similarity according to the computed clusters.

**II.c Entity Ranking.** In the final step, we rank the expanded entity result set for $q$, taking into account similarity to the query and the modelled query type affinity.

## 4 Data Pre-processing and Entity Clustering

In this section, we describe the offline pre-processing to cluster entities and remedy the sparsity of explicit entity links.

## 4.1 Entity Feature Vectors

Entity similarity is measured based on a set of structural and lexical features, denoted by the *entity feature vector* $F(e)$. The features for clustering are described below.

**Lexical Features:** We consider a weighted set of *unigrams* and *bigrams* for an entity $e$, by extracting all textual literals used to describe $e$ denoted as $\mathbf{W}_1(e)$ and $\mathbf{W}_2(e)$. The weights are computed using the standard *tf–idf* metric. Lexical features represent core features when considering the entity retrieval task, more so for the clustering process. A high lexical similarity between an entity pair is a good indicator for expanding the result set from the corresponding cluster space.

**Structural Features:** The feature set $\phi(e)$ considers the set of all object properties that describe $e$. The range of values for the structural features is $\phi(o,e) \rightarrow [0,1]$, i.e., to indicate if a object value is present in $e$. **Feature Space:** To reduce the feature space, we filter out items from the lexical and structural features that occur with low frequency across entities and presumably, have a very low impact on the clustering process due to their scanty occurrence.

## 4.2 Entity Bucketing & Clustering

**Entity Bucketing.** In this step we *bucket* entities of a given *entity type* by computing their *MinHash* signature, which is used thereafter by the LSH algorithm [18]. This step is necessary as the number of entities is very large. In this way we reduce the number of pair-wise comparisons for the entity clustering, and limit it to only the set of entities within a bucket. Depending on the *clustering algorithm*, the impact of bucketing on the clustering scalability varies. Since the LSH algorithm itself has linear complexity, bucketing entities presents a scalable approach considering the size of datasets in our experimental evaluation. A detailed analysis is presented in Section 7.

**Entity Clustering.** Based on the computed feature vectors, we perform entity clustering for the individual entity types and the computed LSH buckets. Taking into account scalability aspects of such a clustering process we consider mainly two clustering approaches: (i) *X–means* and (ii) *Spectral Clustering*. In both approaches we use Euclidean distance as the similarity metric. The dimensions of the Euclidean distance are the feature items in $F(\cdot)$. The similarity metric is formally defined in Equation 1.

$$d(e,e') = \sqrt{\sum (\mathbf{F}(e) - \mathbf{F}(e'))^2} \tag{1}$$

where the sum aggregates over the union of feature items from $\mathbf{F}(e), \mathbf{F}(e')$. The outcome of this process is a set of clusters $\mathbf{C} = \{C_1, \ldots, C_n\}$. The clustering process represents a core part of our approach from which we expand the entity results set for a given query, beyond the entities that are retrieved by a baseline as a starting point. The way the clusters are computed has an impact on the entity retrieval task, thus we present a thorough evaluation of cluster configurations in Section 7.1.

**X–means** To cluster entities bucketed together through the LSH algorithm and of specific entity types, we adopt an extended version of *k-means* clustering, presented by Pelleg et al. which estimates the number of clusters efficiently [16]. *X–means* overcomes two major drawbacks of the standard *k-means* clustering algorithm; (i) computational scalability, and (ii) the requirement to provide the number of clusters $k$ beforehand. It extends the *k–means* algorithm, such that a user only specifies a range $[K_{min}, K_{max}]$ in which the number of clusters, $K$, may reasonably lie in. The bounds for $K$ in our case are set to $[2,50]$ clusters.

**Spectral Clustering** In order to proceed with the *spectral clustering* process, we first construct the adjacency matrix $\mathbf{A}$. The adjacency matrix corresponds to the similarity between entity pairs $d(e,e')$ of a given entity type and bucket. Next, from $\mathbf{A}$ we compute the unnormalised graph Laplacian [22] as defined in Equation 2:

$$\mathbf{L} = diag(\mathbf{A}) - \mathbf{A} \tag{2}$$

where, $diag(\mathbf{A})$ corresponds to the diagonal matrix, i.e., $diag(\mathbf{A})_{i,i} = \mathbf{A}_{i,j}$ for $i = j$.

From matrix $\mathbf{L}$ we are particularly interested in specific properties, which we use for *clustering* and which are extracted from the *eigenvectors* and *eigenvalues* by performing a singular value decomposition on $\mathbf{L}$. The eigenvectors correspond to a square matrix $n \times n$, where each row represents the projected entity into a $n$-dimensional space. Eigenvectors are later used to cluster entities using standard *k–means* algorithm.

However, an important aspect that has impact on the clustering accuracy, is the number of dimensions considered for the *k–means* and the $k$ itself. We adopt a heuristic proposed in [22]. The number of dimensions that are used in the clustering step corresponds to the first spike in the eigenvalue distribution. In addition, this heuristic is also used to determine the number $k$ for the clustering step.

## 5 Entity Retrieval - Expansion and Reranking

In this section, we describe the online process of entity retrieval, including the process of *expansion* and *re-ranking* of the query result set.

### 5.1 Query-biased Results Expansion

Having obtained an initial result set $E_b = \{e_1, \ldots, e_k\}$ through a state of the art ER method (BM25f), the next step deals with expanding the result set for a given user query. From entities in $E_b$, we extract their corresponding set of clusters $\mathbf{C}$ as computed in the pre-processing stage. The result set is expanded with entities belonging to the clusters in $\mathbf{C}$. We denote the entities extracted from the clusters with $E_c$.

There are several precautions that need to be taken into account in this step. We define two threshold parameters for expanding the result set. The first parameter, *cluster size*, defines a threshold with respect to the number of entities belonging to a cluster. If the number is above a specific threshold, we do not take into account entities from that cluster. The underlying rationale is that clusters with a large number of entities tend to be generic and less homogeneous, i.e. they tend to be a weak indicator of similarity. The second parameter deals with the *number of entities* with which we expand the result set for a given entity cluster. The entities are considered based on their distance to the entity $e_b$. We experimentally validate the two parameters in Section 7.

The *fit* of expanded entities $e_c \in E_c$ concerns their similarity to query $q$ and the similarity to $e_b$, which serves as the starting point for the expansion of $e_c$. We measure the *query-biased* entity similarity in Equation 3, where the first component of the equation measures the *string* distance of $e_c$ to $q$, that is $\varphi(q,e_c)$. Furthermore, this is done relative to entity $e_b$, such that if the $e_b$ is more similar to $q$, $\varphi(q,e_b) < \varphi(q,e_c)$ the similarity score will be increased, hence, the expanded entity $e_c$ will be penalized later on in the ranking (note that we measure distance, therefore, the lower the $sim(q,e)$ score the more similar an entity is to $q$).

The second component represents the actual distance score $d(e_b,e_c)$.

$$sim(q,e_c) = \lambda \frac{\varphi(q,e_c)}{\varphi(q,e_b)} + (1-\lambda)d(e_b,e_c) \tag{3}$$

We set the parameter $\lambda = 0.5$, such that entities are scored equally with respect to their match to query $q$ and the distance between entities, based on our baseline approach. The main outcome of this step is to identify possibly relevant entities that have been missed by the scoring function of BM25F. Such entities could be suggested as relevant from the extensive clustering approaches that consider the structural and lexical similarity.

### 5.2 Query Analysis for Re-ranking

Following the motivation example in Figure 1b, an important factor on the re-ranking of the result set is the *query type affinity*. It models the relevance likelihood of a given entity type $t_e$ for a specific query type $t_q$. We give priority to entities that are most likely to be relevant to the the given query type $t_q$ and are least likely to be relevant for other query types $t'_q$. The probability distribution is modeled empirically based on a previous dataset, BTC10. The score $\gamma$, we assign to any entity coming from the expanded result set is computed as in Equation 4.

$$\gamma(t_e,t_q) = \frac{p(t_e|t_q)}{\sum\limits_{t'_q \neq t_q} \left(1 - p(t_e|t'_q)\right)} \tag{4}$$

An additional factor we use in the re-ranking process is the *context score*. To better understand the query intent, we decompose a query $q$ into its *named entities* and additional *contextual terms*. An example is the query $q = \{$*'harry potter movie'*$\}$ from our query set, in which case the contextual terms would be '*movie*' and the named entity '*Harry Potter*' respectively. In case of ambiguous queries, the contextual terms can further help to determine the query intent. The *context score* (see Equation 5) indicates the relevance of entity $e$ to the contextual terms $Cx$ of the query $q$. For entities with a high number of textual literals, we focus on the main literals like *labels, name* etc.

$$context(q,e) = \frac{1}{|Cx|} \sum_{c_x \in Cx} \mathbb{1}_{e \texttt{ has } c_x} \tag{5}$$

### 5.3 Top–*k* Ranking Model

The final step in our entity retrieval approach, re-ranks the expanded entity result set for a query $q$. The result set is the union of entities $\mathbf{E} = \mathbf{E}_b \cup \mathbf{E}_c$. In the case of entities retrieved through the baseline approach $e \in \mathbf{E}_b$, we simply re-use the original score, but normalize the values between $[0,1]$. For entities from $E_c$ we normalize the similarity score relative to the rank of entity $e_b$ (the position of $e_b$ in the result set) which was used to suggest $e_c$. This boosts entities which are the result of expanding top-ranked entities.

$$rank\_score(e) = \begin{cases} \frac{sim(q,e)}{rank(e_b)} & \text{if } e \in E_c \\ bm25f(q,e) & \text{otherwise} \end{cases} \tag{6}$$

The final ranking score $\alpha(e,t_q)$, for entity $e$ and query type $t_q$ assigns higher rank score in case the entity has high similarity with $q$ and its type has high relevance likelihood of being relevant for query type $t_q$. Finally, depending on the query set, in case $q$ contains contextual terms we can add $context(q,e)$ by controlling the weight of $\lambda$ (in this case $\lambda = 0.5$).

$$\alpha(e,t_q) = \lambda\,(rank\_score(e) * \gamma(t_e,t_q)) + (1-\lambda) * context(q,e) \tag{7}$$

The score $\alpha$ is computed for all entities in **E**. In this way based on observations of similar cases in previous datasets, like the BTC10 we are able to rank higher entities of certain types for specific queries.

## 6 Experimental Setup

Here we describe our experimental setup, specifically the datasets, baselines and the ground truth. The setup and evaluation data are available for download[3].

### 6.1 Evaluation Data

**Dataset.** In our experimental setup we use the BTC12 dataset [9]. It represents one of the largest periodic crawls of Linked Data, also containing well-known knowledge bases like Freebase and DBpedia. The overall statistics of the data are: (i) 1.4 billion triples, (ii) 107,967 graphs, (iii) 3,321 entity types, and (iv) 454 million entities.

**Entity Clusters.** The statistics for the generated clusters are as follows: the average number of entities fed into the *LSH* bucketing algorithm is 77,485, whereas the average number of entities fed into *x–means* and *spectral* is 400. The number of generated entity buckets by LSH is 20,2009, while the number of clusters for *x–means* and *spectral* is 13 and 38, with an average of 10 and 20 entities per cluster respectively.

**Query Dataset.** To evaluate our retrieval approach we use the *SemSearch*[4] query set from 2010 with 92 queries. The SemSearch query set is a standard collection for evaluating entity retrieval tasks.

### 6.2 Baseline and State of the Art

**Baseline.** We distinguish between two cases for the original BM25F baseline: (i) $\mathbf{B_t}$ and (ii) $\mathbf{B_b}$. In the first case, we use the *title* or *label* of an entity as a query field, whereas in the second case we use the full *body* of an entity (consisting of all textual literals). The scoring of the fields is performed similar as in [5].

[3] `http://l3s.de/~fetahu/iswc2015/`

[4] `http://km.aifb.kit.edu/ws/semsearch10/`

**State of the art.** We consider the approach proposed in [21] as the state-of-the-art. Similar to their experimental setup, we analyze two cases: (i) **S1** and (ii) **S2**. **S1** expands the entity set from the baseline approach with directly connected entities, and **S2** expands with entities up to the second hop. For further details we refer the reader to [21]. In our experiments, we found that the **S2** did not result in any significant change in performance when compared to **S1**, and we therefore do not report further on **S2**.

**Our approaches.** We analyze two entity retrieval techniques from our approach. The first is based on the *x–means* clustering approach, which we denote by **XM**. The second technique is based on *spectral* clustering and is denoted by **SP**. In both cases, we only expand the result set with entities coming from clusters with a total of ten entities associated with a cluster (see Section 5.1), and finally add only the most relevant entity based on the $sim(q, e_c)$ score.

**BTC indexes.** For the baseline, we generate a Lucene index, where we index entity profiles on two fields `title` and `body` (consisting of all the textual literals of an entity). The second index is an RDF index over the BTC dataset with support for SPARQL queries, for which we use the RDF3X tool [13]. The first index is used for the baseline approach, while the second for the state of the art approach.

## 6.3 Ground Truth for Evaluation of Entity Retrieval

For each query in the *SemSearch2010* query set, we first establish the ground truth through crowdsourcing. Crowdsourced evaluation campaigns for the task of ad-hoc object retrieval have been shown to be reliable [4, 8]. For each of the 92 queries, we pool the top 50 entities retrieved by the various methods, resulting in the top-k pooled entities corresponding to the query. By doing so we generate 4,600 query-entity pairs.

We deploy atomic tasks in order to acquire relevance labels from the crowd for each query-entity pair. We follow the key prescriptions for task design and deployment that emerged from the work of Blanco et al. [4] to build a ground truth. Workers are asked to assess the relevance of each retrieved entity to the corresponding query on a 5-point Likert-type scale[5].

We collect 5 judgements from different workers for each pair to ensure reliable relevance assessments and discernible agreement between workers. This results in a total of 23,000 judgements. The final relevance of an entity is considered to be the aggregated relevance score over the 5 judgements. We assess and compare the performance of the different methods by relying on the ground truth thus generated (see Section 7).

## 6.4 Evaluation Metrics

Evaluation metrics assess the clustering accuracy and the retrieval performance.

**Cluster Accuracy.** As an initial evaluation, we assess the quality of our clusters. From a set of entities belonging to the same cluster, the accuracy is measured as the ratio of entities that *belong together* over the total number of entities in a cluster, where assessments are obtained through crowdsourcing (see Section 7).

[5] *1:Not Relevant*, *2:Slightly Relevant*, *3:Moderately Relevant*, *4:Fairly Relevant* and *5:Highly Relevant*

**Precision.** P@k measures the precision at rank $k$, in our case $k = \{1, \ldots, 10\}$. It is measured as the ratio of retrieved and relevant entities up to rank $k$ over the total number of entities retrieved up to rank $k$.

**Recall.** R@k is measured as the ratio of retrieved and relevant entities up to rank $k$ over the total number of relevant entities up to rank $k$. The total number of relevant entities for a query is determined by the relevance judgements on a large pool of entities.

**Mean Average Precision**. MAP provides an overall precision of a retrieval approach across all considered ranks.

**Normalized Discounted Cumulative Gain**. It takes into account the ranking of entities generated using one of the retrieval approaches and compares it against the ideal ranking in the *ground truth*.

$$nDCG@k = \frac{DCG@k}{iDCG@k} \quad DCG@k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2 i}$$

where $DCG@k$ represents the discounted cumulative gain at rank $k$, and $iDCG@k$ is the ideal $DCG@k$ computed from the *ground truth*.

## 7 Evaluation and Discussion

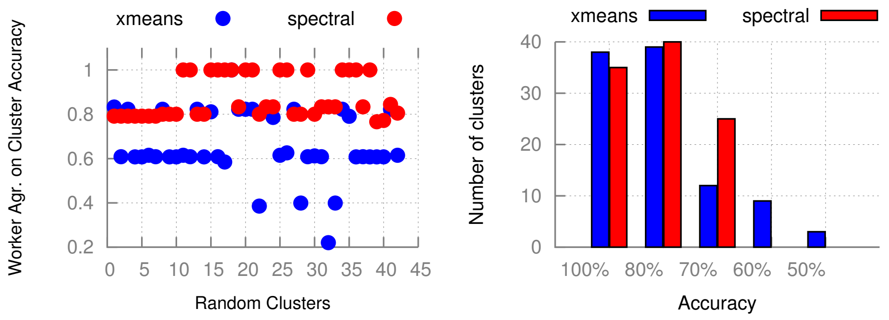
In this section we report evaluation results of the two main steps in our approach. We first evaluate the quality of the pre-processing step, i.e., the clustering results for the *x–means* and *spectral* clustering algorithms. Next, we present the findings from our rigorous evaluation of the entity retrieval task.

### 7.1 Cluster Accuracy Evaluation

Considering the large number of clusters that are produced in the pre-processing step for a given *type* and *bucket*, evaluating the accuracy and quality of all clusters is infeasible. We randomly select 10 entity types and 10 buckets, resulting in 100 clusters for evaluation, where for each cluster we randomly select a maximum of 10 entities.

To evaluate the *cluster accuracy*, we deploy atomic microtasks modeled such that a worker is presented with sets of 10 entities belonging to a cluster, along with a description of the entity in the form of the entity profile. The task of the worker is to pick the odd entities out (if any). We gather 5 judgments from different workers for each cluster. By enforcing restrictions available on the CrowdFlower platform, and following state of the art task design recommendations, we ensure that we receive judgments from the best workers (workers with high reputation as indicated by CrowdFlower).

Figure 3b presents our findings for the evaluation of the clustering process. We note that for *x–means* and *spectral* clustering approaches, nearly 35% and 38% of the clusters are judged to be perfect respectively (i.e., the entities within the cluster were all found to belong together). 39% of the clusters corresponding to *spectral* clustering and 40% of the clusters corresponding to *x-means*, have an accuracy of 80%. Considering its multidimensional representation of the entities, *spectral* clustering has higher accuracy and it does not have clusters below 70% accuracy. The lowest accuracy of 70% for *spectral* clustering implies that in each cluster there were only 3 entities that did not belong to the cluster. The implications of an accurate clustering process become clearer in the next section, where we assess the accuracy of finding relevant entities in the generated entity clusters.



(a) (b)

Fig. 3: (a) Worker agreement on cluster accuracy for *spectral* and *x–means* clustering. (b) Cluster accuracy for the *spectral* and *x–means* clustering approaches.

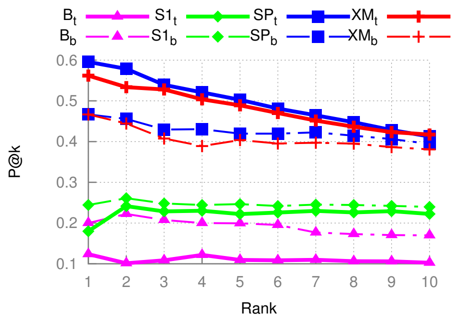
Figure 3a presents the pairwise agreement between workers on the quality of each cluster. In case of the *spectral* clustering, we observe a high inter-worker agreement of 0.75 as per Krippendorf's Alpha. We observe a moderate inter-worker agreement of 0.6 as per Krippendorf's Alpha on the clusters resulting from *x–means*.

## 7.2 Entity Retrieval Evaluation

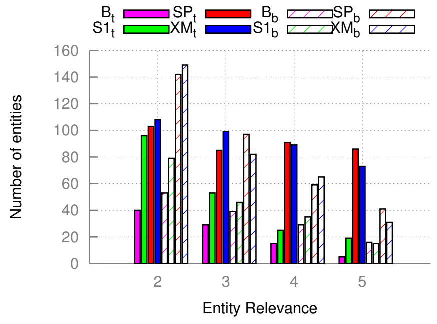
Figure 4a presents a detailed comparison between the $P@k$ for the different methods. The proposed approaches outperform the baseline and state of the art at all ranks. The precision is highest at $P@1 = 0.6$ whereas for the later ranks it stabilizes at 0.4. In contrast to our approach, the performance of the baseline and the state of the art is more uniform, and is around $P@k = 0.25$. The best overall performing approach is the retrieval approach based on spectral clustering *SP*. Table 1 shows the details about the performance of the respective approaches as measured for our evaluation metrics.

An interesting observation is that for our approaches the best performance is achieved when querying for the field *title*. In the case of the baseline, the best performance is achieved when querying for the field *body* ($B_b$) while the same is inconclusive in case of the state-of-the-art methods ($S1_t$ and $S1_b$). We achieve a significantly higher retrieval performance when using the title field. This can be explained by the fact that entities that match a query on their *title* field when compared to those that match a query on their *body* field, have a higher likelihood of being an exact match.

The high gain in performance through our methods (*SP* and *XM*) stems mainly from the two steps in our approach. The first step expands the result set with relevant entities as shown in Figure 4b. The figure shows the number of relevant entities corresponding to the different grading scales as described in Section 7.1. In all cases we note that



(a)



(b)

Fig. 4: (a) P@k for the different entity retrieval approaches under comparison. (b) The relevant entity frequency based on their graded relevance (from *2-Slightly Relevant* to *5-Highly Relevant*) for the different methods.

our methods find more relevant entities. The second step which re-ranks the expanded result set helps in reducing the number of *'non-relevant'* entities. We find that $S1_t$ has a 14% decrease of non-relevant entities, whereas $SP_t$ and $XM_t$ depict a 35% decrease, respectively. In second case where we query the *body* field, the number of *'non-relevant'* entities for $S1_b$ decreases by about 13%, while $SP_b$ and $XM_b$ depict a 24% decrease.

| | $B_t$ | $B_b$ | $S1_t$ | $S1_b$ | $SP_t$ | $SP_b$ | $XM_t$ | $XM_b$ |
|---|---|---|---|---|---|---|---|---|
| P@10 | 0.103 | 0.170 | 0.222 | 0.240 | 0.413 | 0.394 | **0.417** | 0.381 |
| R@10 | 0.052 | 0.089 | 0.112 | 0.118 | 0.206 | **0.219** | 0.216 | 0.215 |
| MAP | 0.110 | 0.191 | 0.224 | 0.246 | **0.497** | 0.426 | 0.482 | 0.407 |
| $Avg(R)$ | 0.031 | 0.058 | 0.063 | 0.074 | 0.132 | **0.133** | 0.131 | 0.130 |

Table 1: Performance of the different entity retrieval approaches. In all cases our approaches are significantly better in terms of P/R ($p < 0.05$ measured for *t-test*) compared to *baseline* and *state of the art*. There is no significant difference between *SP* and *XM* approaches.

We additionally analyze the performance of the entity retrieval approaches through the *NDCG@k* metric. Figure 5 shows the NDCG scores. Similar to our findings for *P@k* presented in Table 1, our approaches perform best for the query field *title* and significantly outperform the approaches under comparison.

Next, we present observations concerning the different *query types* and the entity result set expansion (see Section 5.1) parameters. In Figure 6a we show the improvement we gain in terms of MAP for the different *query types*. We observe that there is quite a variance for the different query types, however, in nearly all cases, the biggest improvement is achieved through the *SP* approach. Interestingly for the query type *'Creative Work'* the state of the art is nearly as good as the *XM* approach, whereas in the case of *'Weapon'* the baseline performs best. One possible explanation for this is that in the case of *'Creative Work'* the explicit entity similarity statements are abundant.

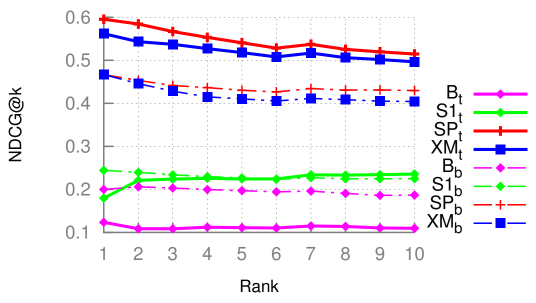
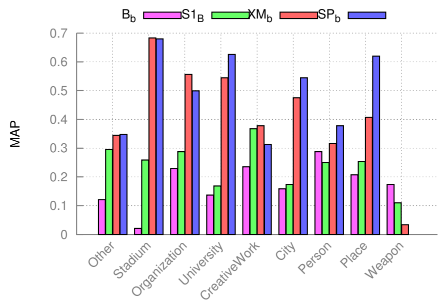
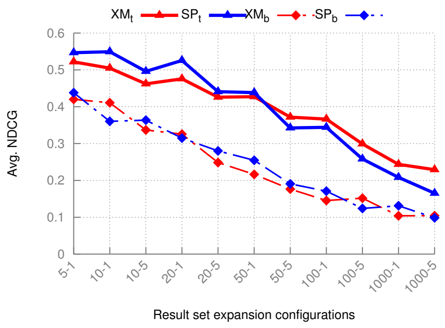


Fig. 5: NDCG@$k$ for $B1$, $S1$ and $SP$, $XM$

Addressing the case of optimizing our retrieval approaches, *SP* and *XM*, we experimentally show the impact that the expansion of the result set has on the measured performance metrics. Here, we show the impact on the *average NDCG* score. Figure 6b shows the performance at average NDCG for the varying *cluster size* and *number of entities* added (result set expansion) for every entity in $E_b$. The best performance is achieved for a rather smaller *cluster size* ranging between 5 and 10 entities per cluster. Regarding the number of entities with which the result set is expanded for every $e_b$, the best performance is achieved by expanding with one entity per cluster. The increase in cluster size and number of entities attributes to a decrease in performance.





(a) (b)

Fig. 6: (a) The aggregated MAP for different query types and for the different retrieval approaches (note, we show the results for field *body* where baseline performs best). (b) The various configurations for the number of expanded entities for *SP* and *XM*.

## 7.3 Discussion

**Scalability.** In the pre-processing stage we introduced the clustering approaches, which first bucket entities together based on the LSH algorithm. This particular step significantly improves the *scalability* of such an offline step. If considering the *x-means* algorithm, under the simplistic assumption that it represents the original *k–means* for which the complexity is $\mathscr{O}(n^{dk+1}log(n))$ (we assume the number of dimensions for the Euclidean space is fixed) for a fixed number of clusters and dimensions. Now, clustering without the bucketing step, we would have around $n = 77,485$ entities for clustering with an average of $k = 13$ clusters. Hence, $\mathscr{O}(77485^{d\cdot 13+1}log(77485)) > \mathscr{O}(400^{d\cdot 13+1}log(400))$, where after bucketing we have on average $n = 400$. Thus, we see a significant decrease in the runtime (while the complexity in theory remains of the same magnitude). For the case of *spectral* clustering this is even more evident, where for the adjacency matrix we consider $n(n-1)/2$ entity pairs, and its singular value decomposition (dependent on the algorithm used) is cubical in terms of big-O notation.

**Crowdsourced Evaluation: Precautions.** In order to ensure that we acquire reliable responses from the crowd workers, we take several precautions while designing the tasks for the evaluation of clusters, as well as establishing the ground-truth for the retrieval of entities. We provide clear instructions and examples to avoid misinterpretations in the relevance scoring, leading to a bias in the judgements. We compensate workers with monetary incentives that are proportionate to their contribution. In addition, we use *gold standard questions* as recommended by previous works to curtail malicious activity.

**Caveats and Limitations.** Considering the optimization of the pre-processing step, the process scales well even for large datasets like the BTC. The retrieval task itself is an online process with no complex approaches and hence the corresponding computational overhead is negligible for the user. We acknowledge the need to re-cluster entities periodically in order to maintain a persistently good entity retrieval performance. However, we believe that this is a relatively minor overhead, when compared to the improvement in performance that it brings about, and given the fact that it is an offline process which can be scaled using parallel infrastructure.

## 8 Conclusions and Future Work

In this work, we presented an approach to improve the performance of entity retrieval on structured data. Building on existing state of the art methods, we follow an approach consisting of offline preprocessing clustering, and online retrieval, results expansion and reranking. Preprocessing exploits *x–means* and *spectral* clustering algorithms using lexical as well as structural features. The clustering process was carried out on a large set of entities (over 450 million). The evaluation of the clustering process shows that over 80% of clusters have an accuracy of more than 80%. As part of the online entity retrieval, for a given a starting result set of entities as retrieved by the baseline approach BM25F we further expand the result set with relevant entities. Additionally, we propose an entity ranking model that takes into account the query type affinity. Finally, we carry out an extensive evaluation of the retrieval process using the SemSearch and the BTC12 datasets. The results show that our methods outperform the baseline and state of the art approaches. In terms of standard IR metrics, our method in combination with one of the clustering approaches, e.g. $SP_t$ improves over $S1_t$ with $\Delta P@10 = +0.19$, $\Delta MAP = +0.273$ and $\Delta R@10 = +0.1$.

***Acknowledgements:*** This work is partly funded by the EU under FP7 project DURAARK (grant no. 600908) and ERC Advanced Grant ALEXANDRIA (grant no. 339233).

## References

1. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th ISWC*, pages 722–735, 2007.
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
3. R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *Proceedings of the 12th ISWC*, pages 33–48, 2013.
4. R. Blanco, H. Halpin, D. M. Herzig, P. Mika, J. Pound, H. S. Thompson, and D. T. Tran. Repeatable and reliable search system evaluation using crowdsourcing. In *Proceeding of the 34th ACM SIGIR*, pages 923–932, 2011.
5. R. Blanco, P. Mika, and S. Vigna. Effective and efficient entity search in rdf data. In *Proceedings of the 10th ISWC*, pages 83–97, Berlin, Heidelberg, 2011. Springer-Verlag.
6. G. Demartini, M. M. S. Missen, R. Blanco, and H. Zaragoza. Entity summarization of news articles. In *Proceeding of the 33rd ACM SIGIR*, pages 795–796, 2010.
7. C. Guéret, P. T. Groth, C. Stadler, and J. Lehmann. Assessing linked data mappings using network measures. In *Proceedings of the 9th ESWC*, pages 87–102, 2012.
8. H. Halpin, D. M. Herzig, P. Mika, R. Blanco, J. Pound, H. S. Thompson, and T. T. Duc. Evaluating ad-hoc object retrieval. In *Proceedings of the IWEST*, 2010.
9. A. Harth. Billion Triples Challenge data set. Downloaded from http://km.aifb.kit.edu/projects/btc-2012/, 2012.
10. Y. Lv and C. Zhai. When documents are very long, bm25 fails! In *Proceedings of 34th ACM SIGIR*, pages 1103–1104, New York, NY, USA, 2011. ACM.
11. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
12. P. Mika, E. Meij, and H. Zaragoza. Investigating the semantic gap through query log analysis. In *Proceedings of the 8th ISWC*, pages 441–455, Berlin, Heidelberg, 2009. Springer-Verlag.
13. T. Neumann and G. Weikum. Rdf-3x: A risc-style engine for rdf. *Proc. VLDB Endow.*, 1(1):647–659, Aug. 2008.
14. Z. Nie, Y. Ma, S. Shi, J.-R. Wen, and W.-Y. Ma. Web object retrieval. In *Proceedings of the 16th WWW*, pages 81–90, 2007.
15. E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52, 2008.
16. D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.
17. J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th WWW*, pages 771–780, 2010.
18. A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.
19. C. B. Robert Meusel, Petar Petrovski. The webdatacommons microdata, rdfa and microformat dataset series. In *The 13th ISWC*, 2014.
20. S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, Apr. 2009.
21. A. Tonon, G. Demartini, and P. Cudré-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *Proceedings of the 35th ACM SIGIR*, pages 125–134, 2012.
22. U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.
23. N. Zhiltsov and E. Agichtein. Improving entity search over linked data by modeling latent semantics. In *Proceedings of the 22nd ACM CIKM*, pages 1253–1256, 2013.