Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

RQUERY: Rewriting Natural Language Queries on Knowledge Graphs to Alleviate the Vocabulary Mismatch Problem

Saeedeh Shekarpour
Kno.e.sis Center
Dayton, United States
saeedeh@knoesis.org

Edgard Marx
AKSW Research Group
Leipzig, Germany
marx@informatik.uni-leipzig.de

Sören Auer
EIS Research Group
Bonn, Germany
auer@cs.uni-bonn.de

Amit Sheth
Kno.e.sis Center
Dayton, United States
amit@knoesis.org

Abstract

For non-expert users, a textual query is the most popular and simple means for communicating with a retrieval or question answering system. However, there is a risk of receiving queries which do not match with the background knowledge. Query expansion and query rewriting are solutions for this problem but they are in danger of potentially yielding a large number of irrelevant words, which in turn negatively influences runtime as well as accuracy. In this paper, we propose a new method for automatic rewriting input queries on graph-structured RDF knowledge bases. We employ a Hidden Markov Model to determine the most suitable derived words from linguistic resources. We introduce the concept of triple-based co-occurrence for recognizing co-occurred words in RDF data. This model was bootstrapped with three statistical distributions. Our experimental study demonstrates the superiority of the proposed approach to the traditional n-gram model.

Introduction

While the amount of information being published on the Web of Data is dramatically high, retrieving information is an issue due to several known challenges. A key persisting challenge is "the lack of accurate knowledge of the vocabulary used", which even expert users frequently use incorrectly. Use of simple interfaces (i.e., textual queries as used by commercial Web search engines) require automatic ways for tackling the *vocabulary mismatch challenge*. This challenge is even more important for schema-aware search systems such as question answering systems rather than schema-unaware search systems such as information retrieval approaches because there precise interpretation of the input query as well as accurate spotting of the answer is more demanding. For instance, using DBpedia as knowledge base, the query ?Who is the designer of Brooklyn Bridge?? could fail, because the desired property in DBpedia is labeled ?architect? and not ?designer?. Key causes of vocabulary mismatching include (i) *Inflectional form*, which is variation of a word for different grammatical categories such as tense, aspect, person, number, etc. For example, the word 'actress' might be required to be altered to 'actor' or 'companies' to 'company'. *Stemming* and *lemmatization* are solutions for reducing this type of mismatching by converting words to a base form. (ii) *Lexical form*, which relates words based on lexical categories. For example, the word 'wife' holds a lexical relation to the word 'spouse' or 'altitude' to 'elevation' because they hold the same meaning.

So far, various approaches have been proposed to address the vocabulary mismatch problem. The most important ones include these. (i) Using a *controlled vocabulary* maintained by human editors. This approach is common for relatively small or restricted domains. (ii) Automatically deriving a *thesaurus*. For instance, word co-occurrence statistics over a text corpus is an automatic way to induce a thesaurus. (iii) *Interactive query expansion*, which provides a list of recommendations for the end user. The recommendations can come from sources such as query logs or thesaurus. (iv) *Automatic query expansion*, which automatically (without any user intervention) adds derived words to the input query in order to increase recall in retrieval systems. (v) Query rewriting based on query log mining, which leverages the manual query rewriting. This approach requires comprehensive query logs, thus being particularly appropriate for web search engines.

Expansion and rewriting methods are endangered to yield large number of irrelevant words which negatively influence runtime as well as accuracy. (Shekarpour et al. 2013) showed that for short queries, the number of derived words is significantly high. Hence we require approaches to heuristically restrict number of derived words and rank them based on their appropriateness. Thus, in this paper, we propose a method for automatic query rewriting on RDF data called **RQUERY**[1]. This method takes into account the topology (i.e. internal structure) as well as the semantics of background knowledge base for query rewriting. Our main contributions are as follows:

- we define the concept of triple-based co-occurrence of words in RDF knowledge bases.
- We extend the Hidden Markov Model for producing and ranking tuples of derived words (i.e. query rewrites).
- We assess and analyze the effectiveness of the proposed approach in two directions:

[1]The source code is available at: https://github.com/shekarpour/RQUERY

1. How effective is the approach for addressing the vocabulary mismatch problem?
2. How effective is the approach for avoiding noise?

In the following section, we present the problem statement and an overview of RQUERY. The next section presents the proposed statistical approach for query rewriting in more detail. Then, we present the results of our experimental study. Next, related work is reviewed. We close with the conclusion and future work.

Problem Statement

Problem Statement and Overview

RDF knowledge base K is regarded as a set of triples $(s, p, o) \in R \times P \times (R \cup L)$, where $R = C \cup I$ is the union of all RDF resources (C, P, I are respectively a set of classes, properties, and instances), and L is the set of literals ($L \cap R = \emptyset$). An RDF knowledge base can be modeled as a graph formally defined as:

Definition 1 (Knowledge Base Graph) *Knowledge base K is modeled as a directed labeled graph $G(V, E)$, where $V = R \uplus L$ is a disjoint union of resources R and literal values L, and $E = P$ is the set of directed edges, where P as properties are edges originating from a resource and ending to either an resource or a literal value.*

The input query $q = (k_1, k_2, ..., k_n)$ is an n-tuple of keywords. With respect to the knowledge base K, the given query q does not have a vocabulary mismatch problem if there is at least one corresponding answer graph for that; otherwise, it contains a vocabulary mismatch problem and is required to be solved. The answer graph roughly is defined as:

Definition 2 (Answer Graph) *The answer graph is a* connected *graph $A = G'(V', E')$ as (i) it is a subgraph out of the knowledge base graph $V' \subset V, E' \subset E$. (ii) the answer of the input query q is embedded in that. (iii) each keyword $k_i \in q$ has a sub-string similarity match on a literal label of either a vertex or an edge.*

If there is no answer graph for the given query q, the input query is rewritten in a way which meets the vocabulary used in the underlying knowledge base. For example, assume the input query is "profession of bandleader". In the vocabulary used in the underlying knowledge base (i.e. DBpedia), instead of the term "profession", the term "occupation" has been used. Thus, the input query should be rewritten as "occupation of bandleader".

Definition 3 (Query Rewrite) *For a given n-tuple query $q = (k_1, k_2, ..., k_n)$, a query rewrite is an m-tuple of keywords $q_r = (k'_1, k'_2, ..., k'_m) | m <= n$ where each k'_i either equals to a keyword $k_i \in q$ or is linguistically derived from a segment $seg_{(x,y)} = (k_x, ..., k_y)$ of the input query q (the minimum length of the segment is 1).*

RQUERY Overview

Our approach, *RQUERY*, obtains an input textual query; then, as output it provides a ranked list of query rewrites q_r which possibly one of them is an appropriate alternative to the original query. Figure 1 illustrates the high level overview of RQUERY, which comprises four main sequential modules. Furthermore, RQUERY relies on external sources, i.e. (1) linguistic thesaurus (e.g. *WordNet* (Fellbaum 1998)) and (2) RDF knowledge base (DBpedia (Lehmann et al. 2009)). In the following, we describe the main objective of each module.

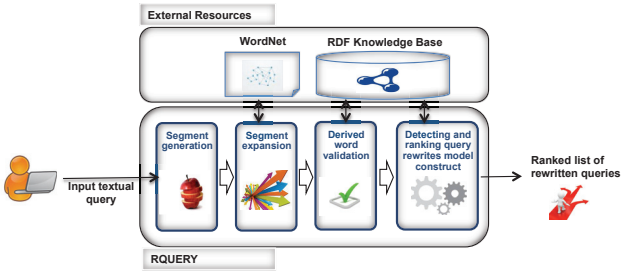


Figure 1: Overview of RQUERY.

I. Segment Generation: An input textual query q is initially preprocessed (i.e. applying tokenization and stop word removal). Then, the remaining keywords are considered as an n-tuple of keywords q. Each subset of this tuple is called a segment. We generate all possible segments which can be derived from q. A segment $seg_{(i,j)}$ is the sequence of keywords from start position i to end position j, $seg_{(i,j)} = (k_i, k_{i+1}, ..., k_j)$. Since the number of keywords is low (in most queries are less than six words[2]), generating this set is not computationally expensive. This set is denoted as $S = \{seg_{(i,j)} | 1 \leq i \leq j \leq n\}$. Furthermore, to limit search space for generating segments, we run a Named Entity (NE) recognizer tool (Finkel, Grenager, and Manning 2005) which finds Named Entities as individual segments. Thus, the remaining keywords, which are not considered NEs, are employed for generating segments with respect to their sequence.

II. Segment Expansion: This module expands segments derived from the previous module using a linguistic thesaurus (please note that this module can be plugged into any linguistic thesaurus such as BabelNet[3], WordNet etc. The linguistic features of WordNet which are employed in RQUERY are (1) *synonyms:* words having the same or a very similar meaning to input word. (2) *hypernyms:* words representing a generalization of the input word. An observation presented in (Shekarpour et al. 2013) reports that basically the *hyponym* (Words representing a specialization of the input word.) relationship leads at deriving a large number of terms whereas their contribution to the vocabulary mismatch task is trivial. Thus, to prevent negative influence on efficiency we exclude them. We initially retrieve the mor-

[2]http://www.keyworddiscovery.com/keyword-stats.html?date=2016-08-01

[3]http://babelnet.org/

phemes set (so as to overcome inflectional forms) for any given segment seg and then expand each morpheme. Thus, for a given segment seg, we define the associated expansion set denoted by ES_{seg} as the union of (1) all morphemes of the given segment seg along with their related forms (2) linguistic words derived via applying linguistic features (i.e. synonym and hypernym) over the existing morphemes of the given segment seg.

III. Derived Word Validation: After constructing the expansion set ES_{seg} for any segment (except NEs), we form the set of all derived words as the union of all available expansion sets $W = \{w | w \in (\bigcup_{\forall seg \in S} ES_{seg})\}$. Each derived word $w \in W$ is validated against the background knowledge base. In other words, we check the occurrence of each word w by sub-string matching with all literals (L) of the underlying RDF knowledge base. Then, simply if no occurrence is observed, the word w is removed from W.

IV. Detecting and ranking possible query rewrites: We aim at distinguishing and ranking possible query rewrites, which respect the intention of the input query q and resolve the vocabulary mismatch of q. In the following section, we discuss that the probability of observing a sequence of words follows Markov property. In addition, since for a given a query rewrite, an original input word is replaced by a derived word which is not directly visible but its output (i.e. original word) is visible; thus, we address the problem of finding the appropriate query rewrite by employing a Hidden Markov Model (HMM).

In the next section we elaborate on all aspects of constructing the model as well as bootstrapping the parameters. Here, we succinctly present an overview. We construct a HMM in three steps:

1. The state space is populated.
2. Transitions between states are established.
3. Parameters are bootstrapped.

Continuing with our running example (i.e."profession of bandleader"); the previous modules of RQUERY derive and validate 10 words for the two given input keywords. The state space is populated with all of these 10 validated words. Then, all the transitions between states are recognised and established. Figure 2 illustrates this model. Each eclipse represents a state containing a derived word. The dashed arrows originating from the states and pointing to the keywords determine the emitted keyword of each state. Transitions between states are represented via black arrows. Arrows originating from the start point indicate states from which the first input keyword is observable. Finally, we run the *Viterbi algorithm* (Viterbi 1967), which is a dynamic programming approach for finding the optimal path through a HMM. This algorithm discovers the most likely states that the sequence of input keywords is observable through. Thus, after running the Viterbi algorithm for the running query *"profession of bandleader"*, the generated top-6 outputs are as follows:

Rank	Probability	Query Rewrite
1	0.0327	profession bandleader.
2	0.0138	profession director.

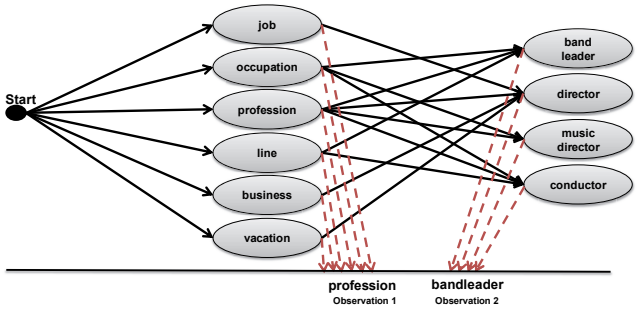


Figure 2: The constructed state space for the query "profession of bandleader".

3	0.0036	profession	conductor.
5	0.00327	profession	music director.
5	0.00327	occupation	bandleader.
6	0.00138	occupation	director.

Statistical Query Rewriting using HMM

According to the *n-gram language model*, the probability of a sequence of n words $W = w_1 w_2 ... w_n$ or w_1^n appearing in the background data is a *joint probability* which can be decomposed as:

$$\mathcal{P}(w_1^n) = \mathcal{P}(w_1)\mathcal{P}(w_2 \mid w_1)...\mathcal{P}(w_n \mid w_{n-1}) =$$

$$\prod_{i=1}^{n} \mathcal{P}(w_i \mid w_1^{i-1})$$

The intuition behind the n-gram model is that instead of computing the probability of a word given its whole preceding subsequence, we can approximate that using only *n* last words. For instance, the *bigram model* (i.e. $n = 2$) approximates $\mathcal{P}(w_n \mid w_1^{n-1})$ as:$\mathcal{P}(w_n \mid w_1^{n-1}) \approx \mathcal{P}(w_n \mid w_{n-1})$. The assumption of the probability of a word depending only on the previous word is known as *Markov property*. The intuitive way to estimate this probability is *Maximum Likelihood Estimation (MLE)* commonly computed by counting subsequences from the corpus. Thus:

$$\mathcal{P}(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_{\forall w} C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

While $C(w_{n-1}w_n)$ is the frequency of the bigram $w_{n-1}w_n$ and $C(w_{n-1})$ is the frequency of the unigram w_{n-1} on background data. In our scenario, we are required to compute the probability of each query rewrite q_r (which is a sequence of words) given an input query q (which is the sequence of input keywords), formally as $\mathcal{P}(q_r \mid q)$, and then rank all possible query rewrites q_r according to their likelihood. Since the RDF data model provides *semantics* as well *structure* to data, thus we aim at computing the likelihood of q_r by taking into account both semantics and structure rather than relying solely on frequency of n-grams. To do that, we model the problem of query rewriting using Hidden Markov Model (HMM). First, we introduce the notation of the HMM parameters, constructing the state space, transition between states, and then we detail how we bootstrap the

parameters of our HMM. Formally, a HMM is a quintuple $\lambda = (X, Y, A, B, \pi)$ where:

- X is a finite set of states. In our case, X equals the set of the validated derived words W. In other words, each word $w \in W$ forms a state.
- Y denotes the set of observations. Here, Y equals the set of all segments $\forall seg \in S$ derived from the input n-tuple of keywords q.
- $A : X \times X \rightarrow [0, 1]$ is the transition matrix. Each entry a_{ij} is the transition probability $\mathcal{P}(S_j|S_i)$ from state S_i to state S_j.
- $B : X \times Y \rightarrow [0, 1]$ represents the emission matrix. Each entry $b_{i-seg} = \mathcal{P}(seg|S_i)$ is the probability of emitting the segment seg from the state S_i.
- $\pi : X \rightarrow [0, 1]$ denotes the initial probability of states.

We define the basic problem as follows: the sequence of input keywords q and the model λ are given, and the problem is to find the *optimal* sequence of states $q_r = (S_1, S_2, ..., S_m)$ which explain the given observation, i.e. input query $q(k_1, ..., k_n)$. Please note that there are possibly multiple distinct sequences of states which the given input query q is observable through, thus the aim is obtaining the optimal one; formally as: $\gamma = \arg\max_{q_r}\{\mathcal{P}(q_r \mid q, \lambda)\}$. $\mathcal{P}(q_r \mid q, \lambda)\}$ is the probability of observing the given query q through the sequence of states q_r. For computing the probability of any query rewrite q_r, the model λ plays a role as a constant parameter, thus we assume

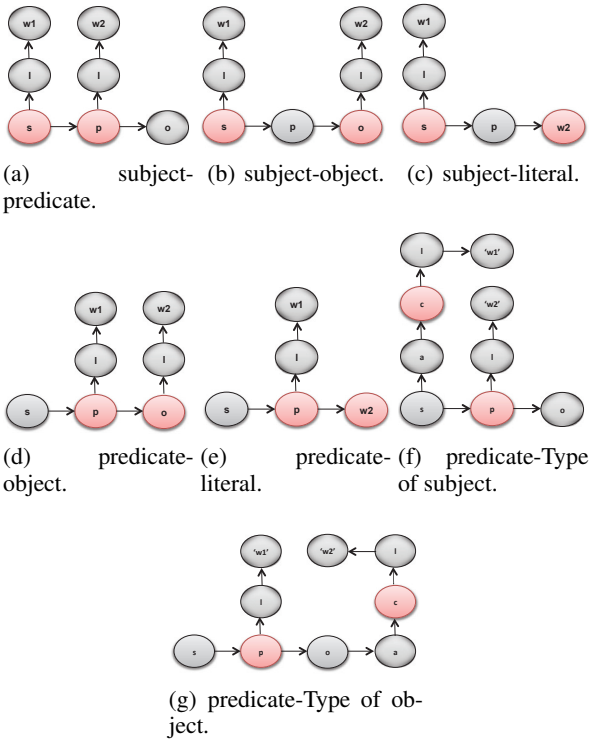
$$\mathcal{P}(q_r \mid q, \lambda)\} \approx \mathcal{P}(q_r \mid q) \Longrightarrow \gamma = \arg\max_{q_r}\{\mathcal{P}(q_r \mid q)\}$$

Assuming that q_r is a sequence of states $(S_1...S_m)$ (please note that each state S_i corresponds to the word w_i). We expand $\mathcal{P}(q_r \mid q)$ as $\mathcal{P}(q_r \mid q) = \mathcal{P}(S_1...S_m \mid k_1...k_n)$. The probability of observing the keyword k_i from the state S_j is denoted as $\mathcal{P}(k_i \mid S_j)$. As from a state S_i either one or multiple keywords might be observable, the number of states m is minor or equal to the number of keywords $m <= n$. Regarding the Markov property, the probability of reaching the state S_m and observing the keyword k_n is equal to $\mathcal{P}(S_m \mid S_{m-1}) * \mathcal{P}(k_n \mid S_m)$. Thus, the equation (2) can be rewritten as:$(\mathcal{P}(S_m \mid S_{m-1}) * \mathcal{P}(k_n \mid S_m)) * \mathcal{P}(S_1...S_{m-1} \mid k_1...k_{n-1})$. It can be extended further as:

$$\alpha * \prod_{i=2,j=2}^{m,n} \mathcal{P}(S_i \mid S_{i-1}) * \mathcal{P}(k_j \mid S_i)|\alpha = \mathcal{P}(S_1) * \mathcal{P}(k_1 \mid S_1)$$

State Space. A-priori, the state space is populated with as many states as the total number of words existing in *literal positions* of triples available in the underlying RDF knowledge base. With this regard, the number of states is thus potentially large. To reduce the number of states, we limit the state space X to the set of *validated derived words* W. Thus, a relevant state is a state for which its associated word w' exists in the derived word set $w' \in W$. From each state, the observable keyword (i.e., emitted strings) is the segment s of which the associated word w of a states is derived. For instance, the word `job` is derived from the segment `profession`, so the keyword `profession` is emitted from the state associated with the word `job`.

Transitions between States. We define transitions between states based on the concept of *co-occurrence of words*. We adopt the concept of co-occurrence of words from the traditional information retrieval context and move it to the RDF knowledge bases. *Triple-based co-occurrence* means co-occurrence of words in literals found in the resource descriptions of the two resources of a given triple:



(a) subject-predicate. (b) subject-object. (c) subject-literal.

(d) predicate-object. (e) predicate-literal. (f) predicate-Type of subject.

(g) predicate-Type of object.

Figure 3: The graph patterns employed for recognising co-occurrence of the two given words $w1$ and $w2$. Please note that the letters s, p, o, c, l and a respectively stand for subject, predicate, object, class, rdfs:label and rdf:class.

1. Two words w_1 and w_2 co-occur in literal values of the property `rdfs:label` of resources placed in the (i) *subject* as well as *predicate* of a given triple (Figure 3(a) shows *subject-predicate* co-occurrence). (ii) *subject* as well as *object* of a given triple ((Figure 3(b) shows *subject-object* co-occurrence). (iii) *predicate* as well as *object* of a given triple (Figure 3(d) shows *predicate-object* co-occurrence).
2. Two words w_1 and w_2 co-occur in the literal of a given triple as well as with the property `rdfs:label` of the resource placed in the (i) *subject* of that triple (Figure 3(c) shows *subject-literal* co-occurrence). (ii) *predicate* of that triple (Figure 3(e) shows *predicate-literal* co-occurrence).
3. Two words w_1 and w_2 co-occur in the literal of the property of the given triple and in the `rdfs:label` of the type of resource placed in the (i) *subject* of that triple (Figure 3(f) shows *type of subject-predicate* co-occurrence). (ii) *object* of that triple (Figure 3(g) shows *type of object-*

predicate co-occurrence).

Bootstrapping Parameters. Commonly, supervised learning is employed for estimating the Hidden Markov Model parameters. An important consideration here is that we encounter a dynamic modelling meaning state space as well as issued observation (i.e., sequence of input keywords) vary query by query. Thus, learning probability values should be generic and not query-dependent because learning model probabilities for each individual query is not feasible. Instead, we rely on *bootstrapping*, a technique used to estimate an unknown probability distribution function. We apply three distributions (i.e., normal, uniform and zipfian) to find out the most appropriate distribution.

For bootstrapping the model parameters A and π, we take into account *co-occurence* between words as well as *frequency of words* (denoted by w_f) in the RDF knowledge base. Word frequency is generally defined as the number of times a word appears in the knowledge base. Herein, we adopt an implicit word frequency which prioritizes resources based on their type. In our previous research (Shekarpour et al. 2011), we observed that generally the frequency of resources with type *class* and *property* is higher than instance resources. In DBpedia, for example, classes have an average frequency of 14,022, while properties have on average 1,243 and instances 37. We assign a static word frequency, based on word appearance position. In other words, if a given word w appears in the label of a class, it obtains higher word frequency value. The equation 1 specifies the word frequency values according to their appearance position. With respect to our underlying knowledge base (i.e., DBpedia), we assign *logarithm* of the average of frequency for each α; for instance, $\alpha 2$ approximates $log(1243) \sim 3$.

$$w_f = \begin{cases} \alpha_1 = \log \overline{Class_f} & w \in ClassLabels \\ \alpha_2 = \log \overline{Property_f} & w \in PropertyLabels \\ \alpha_3 = \log \overline{Instance_f} & w \in InstanceLiterals \end{cases} \quad (1)$$

Transition Probability. The transition probability of state S_j following state S_i is denoted as $a_{ij} = \mathcal{P}(S_j|S_i)$. Note that the condition $\sum_{S_i} \mathcal{P}(S_j|S_i) = 1$ holds. Here, the $S_i(w_f)$ is the word frequency of the derived word associated to the state S_i. The probabilities from state S_i to the neighbouring states are uniformly distributed based on the frequency values. Consequently, states with higher frequency values are more probable to be met. The transition probability from the state S_i to the state S_j is computed by:

$$a_{ij} = \mathcal{P}(S_j \mid S_i) = \frac{S_j(w_f)}{\sum_{\forall k, a_{ik}>0} S_k(w_f)}$$

Initial Probability. The initial probability $\pi(S_i)$ is the probability that the model assigns to the initial state S_i at the beginning. The initial probabilities fulfill the condition $\sum_{\forall S_i} \pi(S_i) = 1$. We denote states for which the first keyword is observable by $InitialStates$. In fact, $\pi(S_i)$ of an initial state is uniformly distributed on frequency values. The initial probabilities are defined as follows:

$$\pi(S_i) = \frac{S_i(w_f)}{\sum_{\forall S_j \in InitialStates} S_j(w_f)}$$

Emission Probability. The probability of emitting a given segment seg from the state S_i depends on the linguistic relation between the word associated to that state $S_i(w)$ and the given segment seg. The following equation represents the likelihood of observing the segment seg of the input query q through the given word w which is associated to the state S_i. In this equation, $\delta(seg, S_i(w))$ measures syntactic and semantic distance between the given segment seg and the word w associated to the state S_i. This distance is computed based on levenshtein edit distance and semantic distance derived from WordNet. Thus, for each given segment and state, the emission probability is computed as:

$$b_{ik} = \mathcal{P}(seg|S_i(w)) = exp(\delta(seg, S_i(w)))$$

Evaluation.

The goal of our evaluation is investigating positive as well as negative impacts of the proposed approach by raising the following two questions: (1) How effective is the approach for addressing the vocabulary mismatch problem when employing queries having a vocabulary mismatch problem?; (2) How effective is the approach for avoiding noise when employing queries which do not have a vocabulary mismatch problem? We employ *Mean Reciprocal Rank (MRR)* (Voorhees and Tice 1999) as our metric for measuring accuracy of the outputs. Please note that since all our computations are carried out on the fly; thus, in this work, runtime is not the subject of our evaluation. Yet, there is no standard benchmark for query rewriting task on RDF knowledge bases. However, a schema-agnostic challenge[4] was presented at the European Semantic Web Conference (ESWC) in 2015 which provides an evaluation test collection for schema-agnostic query mechanisms on RDF datasets (i.e. DBpedia). In this benchmark, each query is annotated with the required mappings to the background knowledge base. Not all of the provided queries suit our query rewriting scenario, because there is not always a one-to-one transformation between the input and output keywords. In other words, in some queries a non-stop word of the input query is ignored ($w \rightarrow null$) or in some queries a non-stop word keyword is added to the query ($null \rightarrow w$). For instance, for the given query `countries in which the Yenisei river flow through` the input keyword `flow` is ignored. We excluded all the queries which do not have a one-to-one transformation. Then, we prepared a training collection (containing 40 queries) and a test collection (containing 37 queries) used respectively for the bootstrapping and the final evaluation. Our query collection is provided as supplementary material.

[4] https://sites.google.com/site/eswcsaq2015/documents

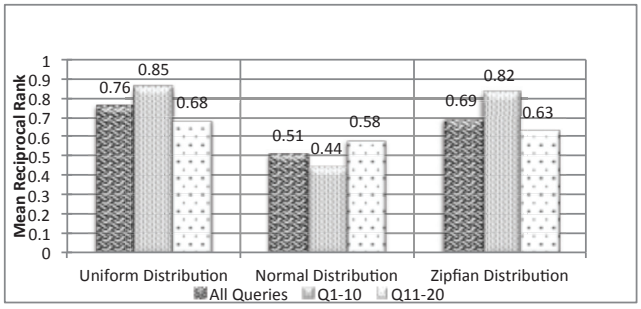


Figure 4: Mean reciprocal rank for applying uniform, normal and zipfian distributions.

Evaluation of Bootstrapping. We perform an experimental study to discover the optimum setting for initial and transition probabilities. Figure 4 represents the achieved MRR for bootstrapping the model by employing three distributions, i.e., uniform, normal, and zipfian distributions. This graph diffrentiates MRR for three categories of queries (i) all queries (Q1-20), (ii) queries having a vocabulary mismatch problem (Q1-10), and (iii) queries which do not have a vocabulary mismatch problem (Q11-20). As can be observed in Figure 4, uniform distribution outperforms normal and zipfian distribution for all three categories of queries. The achieved MRR value (especially for the queries which have a vocabulary mismatch problem 0.85) means that the desired query rewrite is mostly placed in the first position. However, even in the case of the third category, the MRR greater than 0.5 (i.e. 0.68) shows that the desired query rewrite captures the top ranks (i.e. rank 1 or 2) in the output of the model.

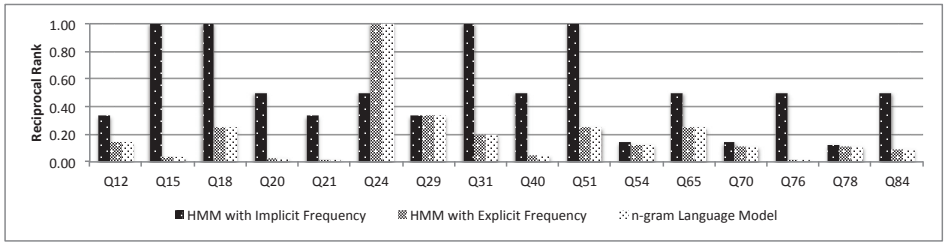
Evaluation of the approach. Our baseline is calculating the probability of a query rewrite q_r using an n-gram language model which is based on the frequency of n-grams in the corpus. Obviously, expecting the occurrence of an n-gram in a single literal in RDF data is not rational; thus, we extend the window for n-gram occurrence to all terms associated to an individual triple. Furthermore, we run the HMM both with implicit as well as explicit frequency of words which were used for calculating transition and initial probabilities. Figure 5 shows the observed Reciprocal Rank (RR) for all the employed queries in the test collection. For queries which have a vocabulary mismatch problem (Figure 5(a)), the achieved MRR for running the Hidden Markov Model based on implicit frequency is 0.53. Thus, the desired query rewrite is placed in the rank one or two. This run outperforms the other two runs with the achieved MRR 0.13 and 0.34, respectively, for running the Hidden Markov Model based on explicit frequency and an n-gram language model. For queries which do not have a vocabulary mismatch problem (Figure 5(b)), the achieved MRR for running the Hidden Markov Model based on implicit frequency is 0.64. This run also outperforms the other two runs with the achieved MRR 0.43 and 0.42, respectively, for running the Hidden Markov Model based on explicit frequency and an n-gram language model. To sum up, the first key conclusion of the experiment is that prioritizing the type of resources in which a word is occurred is effective for computing the word frequency. The second conclusion is that structure as well as semantics of RDF data is well-captured by HMM for rewriting query task.

Related Work

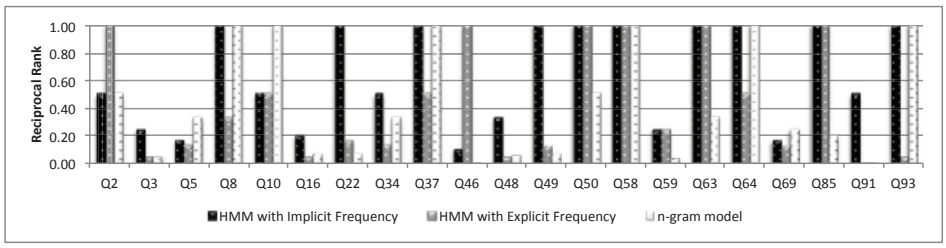
Studies on query rewriting (or expansion) dates from the early 60s (Doyle 1961) and it is a long-standing issue in Information Retrieval (IR). There is a vast literature on Iterative Query Rewriting (IQR) (e.g. (Efthimiadis 1993; Baeza-Yates and Ribeiro-Neto 1999)) which is mostly a manual way for altering query. Previously, Automatic Query Rewriting (AQR) utilized local context provided by the query and was based on pre-processing of top retrieved documents in order to filter false positives before using term-ranking methods. Besides web snippets, several methods for finding more compact and informative document representation has been proposed. Examples of such methods are passage extraction (Xu and Croft 2000), text summarization (Lam-Adesina and Jones 2001) as well as more compact structures such as orthogonal features (Chang, Ounis, and Kim 2006).

Concept terms (Qiu and Frei 1993) and term clustering (Crouch and Yang 1992; Schütze and Pedersen 1995; Bast, Majumdar, and Weber 2007) are two classical corpus specific strategies for AQR. Those approaches rely on probabilistic methods based on the co-occurrence of terms in a document collection. Other approaches explore the corpus with context vectors (Gauch, Wang, and Rachakonda 1999), mutual information (Hu, Deng, and Guo 2006), latent semantic indexing (Park and Ramamohanarao 2007), and interlinked Wikipedia articles (Milne, Witten, and Nichols 2007). Search logs may also encode implicit relevance feedback; other techniques consist of finding queries associated with the same documents (Billerbeck 2005) as well as extracting terms directly from clicked results (Riezler et al. 2007). (Carpineto and Romano 2007) provide a detailed study of automatic query rewriting in information retrieval. The use of an ontology for query rewriting is another well-known AQR technique. In this regard, domain-specific and domain-independent ontologies have been extensively used.

Some related work has been focused on the use of WordNet; however its use may raise several issues; e.g., lack of proper nouns. (Navigli and Velardi 2003) argued that extracting concepts belonging to the same semantic domain of the query is a better approach than the simple use of synonyms. (Mandala, Takenobu, and Hozumi 1998) proposed enriching the WordNet with an automatically constructed thesaurus. (Bhogal, Macfarlane, and Smith 2007) provide a comprehensive study of AQR using ontologies. More recently, there were several methods used which apply AQR on Semantic Web. (Zhang et al. 2013) describe an approach for mining equivalent relations from Linked Data that relies on three measures of equivalency: triple overlap, subject agreement, and cardinality ratio. Existing semantic search engines also employ AQE. For instance, Alexandria (Wendt, Gerlach, and Düwiger 2015) uses Freebase to include synonyms and different surface forms.



(a) Queries which do not have a mismatch problem.



(b) Queries which have a mismatch problem.

Figure 5: Reciprocal Rank for running Hidden Markov Model based on implicit and explicit frequency and also n-gram language model.

MHE[5] combines query rewriting and entity recognition by using textual references to a concept and linking to Wikipedia. Eager (Gunes et al. 2012) rewrites a set of resources with the same type using DBpedia and Wikipedia categories. PowerAqua (Lopez et al. 2012) is an ontology-based question answering system that answers natural language queries and uses WordNet synonyms and hypernyms as well as resources related to the `owl:sameAs` property. In this work, our rewriting method is enhanced by taking the structure as well as semantics of background knowledge into account.

Conclusion and Future work

In this paper, we presented a method for automatic query rewriting. The proposed approach employs the graph structure as well as semantics of RDF data for recognizing and ranking the possible query rewrites. It uses a Hidden Markov Model which its transitions between states defined based on the concept of triple-based co-occurrence. An experimental study was performed on a training dataset in order to detect the optimum distribution for bootstrapping the model parameters. The result of the evaluation shows high accuracy (i.e. high mean reciprocal rank) of the proposed approach in comparison to traditional n-gram language model. We plan to extend this work in several directions. We plan to take into account semantic relations of Linked Data (e.g. `owl:sameAs`) in addition to linguistic features. Furthermore, since all computations are carried out on the fly, in the future we are going to construct a semantic index on co-occurrence of words in order to speed up retrieval requests. Another extension is about extending our benchmark by including more number of queries from various schemes.

[5]http://ups.savba.sk/ marek

Acknowledgments. We acknowledge partial support from the National Science Foundation (NSF) awards: (1) EAR 1520870: Hazards SEES: Social and Physical Sensing Enabled Decision Support for Disaster Management and Response. (2) CNS 1513721: Context-Aware Harassment Detection on Social Media. Any opinions, findings, and conclusions/recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

Baeza-Yates, R. A., and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Bast, H.; Majumdar, D.; and Weber, I. 2007. Efficient interactive query expansion with complete search. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*.

Bhogal, J.; Macfarlane, A.; and Smith, P. 2007. A review of ontology based query expansion. *Information processing & management*.

Billerbeck, B. 2005. *Efficient query expansion*. PhD dissertation, Computer Science and Information Technology, RMIT University.

Carpineto, C., and Romano, G. 2007. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*

Chang, Y.; Ounis, I.; and Kim, M. 2006. Query reformulation using automatically generated query concepts from a document space. *Information processing & management*.

Crouch, C. J., and Yang, B. 1992. Experiments in automatic statistical thesaurus construction. In *Proceedings of the 15th*

annual international ACM SIGIR conference on Research and development in information retrieval.

Doyle, L. B. 1961. Semantic road maps for literature searchers. *Journal of the ACM (JACM)*.

Efthimiadis, E. N. 1993. A user-centred evaluation of ranking algorithms for interactive query expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, 146–159.

Fellbaum, C., ed. 1998. *WordNet: an electronic lexical database*. MIT Press.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Gauch, S.; Wang, J.; and Rachakonda, S. M. 1999. A corpus analysis approach for automatic query expansion and its extension to multiple databases. *ACM Transactions on Information Systems (TOIS)*.

Gunes, O.; Schallhart, C.; Furche, T.; Lehmann, J.; and Ngomo, A.-C. N. 2012. Eager: extending automatically gazetteers for entity recognition. In *Proceedings of the 3rd Workshop on the People's Web Meets NLP: Collaboratively Constructed Semantic Resources and their Applications to NLP*.

Hu, J.; Deng, W.; and Guo, J. 2006. Improving retrieval performance by global analysis. In *18th International Conference on Pattern Recognition (ICPR'06)*.

Lam-Adesina, A. M., and Jones, G. J. 2001. Applying summarization techniques for term selection in relevance feedback. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*.

Lehmann, J.; Bizer, C.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics* 7(3):154–165.

Lopez, V.; Fernández, M.; Motta, E.; and Stieler, N. 2012. Poweraqua: Supporting users in querying and exploring the semantic web. *Semant. web* 3(3):249–265.

Mandala, R.; Takenobu, T.; and Hozumi, T. 1998. The use of wordnet in information retrieval. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, 31–37.

Milne, D. N.; Witten, I. H.; and Nichols, D. M. 2007. A knowledge-based search engine powered by wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*.

Navigli, R., and Velardi, P. 2003. An analysis of ontology-based query expansion strategies. In *Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*.

Park, L. A., and Ramamohanarao, K. 2007. Query expansion using a collection dependent probabilistic latent semantic thesaurus. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Qiu, Y., and Frei, H.-P. 1993. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93.

Riezler, S.; Vasserman, A.; Tsochantaridis, I.; Mittal, V.; and Liu, Y. 2007. Statistical machine translation for query expansion in answer retrieval. In *Annual Meeting-Association For Computational Linguistics*.

Schütze, H., and Pedersen, J. O. 1995. Information retrieval based on word senses. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, 161–175.

Shekarpour, S.; Auer, S.; Ngonga Ngomo, A.-C.; Gerber, D.; Hellmann, S.; and Stadler, C. 2011. Keyword-driven sparql query generation leveraging background knowledge. In *International Conference on Web Intelligence*.

Shekarpour, S.; Höffner, K.; Lehmann, J.; and Auer, S. 2013. Keyword query expansion on linked data using linguistic and semantic features. In *7th IEEE International Conference on Semantic Computing, September 16-18, 2013, Irvine, California, USA*.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.

Voorhees, E. M., and Tice, D. M. 1999. The trec-8 question answering track evaluation. In *In Text Retrieval Conference TREC-8*, 83–105.

Wendt, M.; Gerlach, M.; and Düwiger, H. 2015. *Linguistic Modeling of Linked Open Data for Question Answering*. Berlin, Heidelberg: Springer Berlin Heidelberg. 102–116.

Xu, J., and Croft, W. B. 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems (TOIS)*.

Zhang, Z.; Gentile, A. L.; Augenstein, I.; Blomqvist, E.; and Ciravegna, F. 2013. Mining equivalent relations from linked data. In *ACL*.