Intent-Aware Semantic Query Annotation

Rafael Glater
CS Dept, UFMG
Belo Horizonte, MG, Brazil
rafaelglater@dcc.ufmg.br

Rodrygo L. T. Santos
CS Dept, UFMG
Belo Horizonte, MG, Brazil
rodrygo@dcc.ufmg.br

Nivio Ziviani
CS Dept, UFMG & Kunumi
Belo Horizonte, MG, Brazil
nivio@dcc.ufmg.br

ABSTRACT

Query understanding is a challenging task primarily due to the inherent ambiguity of natural language. A common strategy for improving the understanding of natural language queries is to annotate them with semantic information mined from a knowledge base. Nevertheless, queries with different intents may arguably benefit from specialized annotation strategies. For instance, some queries could be effectively annotated with a single entity or an entity attribute, others could be better represented by a list of entities of a single type or by entities of multiple distinct types, and others may be simply ambiguous. In this paper, we propose a framework for learning semantic query annotations suitable to the target intent of each individual query. Thorough experiments on a publicly available benchmark show that our proposed approach can significantly improve state-of-the-art intent-agnostic approaches based on Markov random fields and learning to rank. Our results further demonstrate the consistent effectiveness of our approach for queries of various target intents, lengths, and difficulty levels, as well as its robustness to noise in intent detection.

CCS CONCEPTS

•**Information systems** →**Retrieval effectiveness;**

KEYWORDS

Semantic query annotation; Learning to rank; Intent-aware;

ACM Reference format:
Rafael Glater, Rodrygo L. T. Santos, and Nivio Ziviani. 2017. Intent-Aware Semantic Query Annotation. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080825

1 INTRODUCTION

A user's search query has traditionally been treated a short, underspecified representation of his or her information need [22]. Despite the trend towards verbosity brought by the popularization of voice queries in modern mobile search and personal assistant interfaces [20], query understanding remains a challenging yet crucial task for the success of search systems. One particularly effective strategy for improving the understanding of a query is to annotate it with semantic information mined from a knowledge base, such as DBPedia.[1] In particular, previous analysis has shown that over 70% of all queries contain a semantic resource (a named entity, an entity type, relation, or attribute), whereas almost 60% have a semantic resource as their primary target [32].

State-of-the-art semantic query annotation approaches leverage features extracted from the descriptive content of candidate semantic resources (e.g., the various textual fields in the description of an entity [28, 44]) or their structural properties (e.g., related semantic resources [39]) in a knowledge base. In common, these approaches treat every query uniformly, regardless of its target intent.[2] In contrast, we hypothesize that queries with different intents may benefit from specialized annotation strategies. For instance, some queries could be effectively annotated with a single entity (e.g., "*us president*") or an entity attribute (e.g., "*us president salary*"). Other queries could be better represented by a list of entities of a single type (e.g., "*us presidents*") or of mixed types (e.g., "*us foreign affairs*"). Finally, some queries may be simply ambiguous and demand annotations suitable for disambiguation (e.g., "*us*").

In this paper, we propose a framework for learning semantic annotations suitable to the target intent of each individual query. Our framework comprises three main components: (i) intent-specific learning to rank, aimed to produce ranking models optimized for different intents; (ii) query intent classification, aimed to estimate the probability of each query conveying each possible intent; and (iii) intent-aware ranking adaptation, aimed to promote the most relevant annotations given the detected intents. To demonstrate the applicability of our framework, we experiment with a state-of-the-art learning to rank algorithm for intent-specific learning, multiple classification approaches for intent classification, and two adaptive strategies for annotation ranking. Thorough experiments using a publicly available semantic annotation test collection comprising queries with different intents show that our proposed framework is effective and significantly improves state-of-the-art intent-agnostic approaches from the literature. Moreover, a breakdown analysis further reveals the consistency of the observed gains for queries of various target intents, lengths, and difficulty levels, as well as the robustness of the framework to noise in intent detection.

In summary, our main contributions are three-fold:

- An intent-aware framework for learning semantic query annotations from structured knowledge bases.
- An analysis of the specificity of several content and structural features for different query intents.
- A thorough validation of the proposed framework in terms of annotation effectiveness and robustness.

SIGIR '17, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080825

[1] http://wiki.dbpedia.org/
[2] While the word "intent" has been used as a synonym of "information need" in some contexts, here we adopt the more traditional definition of intent as the type (or class) of an information need, such as informational or navigational [9].

In the remainder of this paper, Section 2 discusses related work on semantic query annotation and intent-aware information retrieval. Section 3 describes the various components of our proposed framework and their instantiation. Sections 4 and 5 describe the setup and the results of our empirical evaluation. Finally, Section 6 provides our conclusions and directions for future research.

2 RELATED WORK

In this section, we provide an overview of related work on semantic query annotation using knowledge bases. In addition, we discuss related attempts to exploit query intents in different search tasks.

2.1 Semantic Query Annotation

Semantic search approaches [4] have been extensively researched in recent years, motivated by a series of related workshops and evaluation campaigns [1, 2, 15]. While some research has been devoted to semantic search on the open Web [2, 10, 37], particularly relevant to this paper are approaches focused on ranking semantic resources (e.g., named entities) mined from a structured domain, such as a knowledge base. The top ranked resources can be used directly to enrich a search engine's results page with structured semantic information [5] or indirectly to annotate the user's query for further processing for improved search quality.

Search in knowledge bases is typically performed using structured query languages such as SPARQL.[3] However, producing structured queries requires some expertise from the user, which limits the applicability of this approach in a broader scenario. To support unstructured querying, most previous semantic search approaches adapt traditional IR techniques to find, in the knowledge base, resources that match the user's query. For instance, some related works have used standard bag-of-words models, like BM25 [3, 31, 39] and language models (LM) [16, 17, 21, 27, 43]. Extending traditional bag-of-words models, multi-fielded approaches have been proposed to appropriately weight information present in different fields describing a semantic resource. For instance, approaches based on BM25F [6, 7, 12, 18, 31, 39] permit the combination of the BM25 scores of different fields into the final retrieval score. Multi-fielded approaches based on a mixture of language models have also been proposed [11, 29], which linearly combine query likelihood estimates obtained from multiple fields.

Also contrasting with bag-of-words models, recent approaches have exploited dependencies among query term occurrences in the descriptive content of a semantic resource. Building upon the framework of Markov random fields (MRF) [26], these approaches construct a graph of dependencies among the query terms, which is used to estimate the relevance of each retrieved semantic resource. In particular, Zhiltsov et al. [44] introduced a multi-fielded extension of MRF, called FSDM, which estimates the weight of each field with respect to three types of query concept: unigram, ordered bigram, and unordered bigram. FSDM was later extended by Nikolaev et al. [28], who proposed to estimate field weights with respect to individual query concepts. To cope with the explosive number of concepts (i.e., every possible unigram, ordered, and unordered bigram), they instead learn field weights with respect to a fixed set of concept features (e.g., the probability of occurrence of the concept in a field). In contrast to both of these approaches, we propose to learn the appropriateness of intent-specific feature-based ranking models for each individual query, by automatically predicting the target intent of this query. In Section 5, we compare our approach to FSDM as a representative of the current state-of-the-art.

In addition to exploiting the descriptive content of semantic resources, other researchers have adopted a hybrid approach [11, 17, 21, 34, 39], leveraging structural properties of the knowledge base. In these approaches, an initial ranking of semantic resources is either re-ranked or expanded using the knowledge base structure to find related resources, which can be done through structured graph traversals [39] or random walks [34]. For instance, Tonon et al. [39] exploited entities initially retrieved using BM25 as seeds in the graph from which related entities could be reached. Bron et al. [11] proposed a method that makes a linear combination of the scores of a content-based approach using language models and a structure-based approach, which captures statistics from candidate entities represented according to their relations with other entities, expressed in RDF triples. Relatedly, Elbassouni et al. [17] proposed a language modeling approach to rank the results of exact, relaxed, and keyword-augmented graph-pattern queries over RDF triples into multiple subgraphs. The Kullback-Leibler divergence between the query language model and the language models induced by the resulting subgraphs was then used to produce the final ranking. While our main focus is on learning strategies rather than on specific features, to demonstrate the flexibility of our proposed framework, we exploit multiple structural properties of each semantic resource as additional features. In particular, these features are used for both detecting the intent of a query as well as for ranking semantic resources in response to this query.

2.2 Exploiting Query Intents

The intent underlying a user's search query has been subject of intense research in the context of web search. Broder [9] proposed a well-known intent taxonomy, classifying web search queries into informational, navigational and transactional. Rose and Levinson [35] later extended this taxonomy to consider more fine-grained classes. In the context of semantic search, Pound et al. [32] categorized queries into four major intents: entity queries, which target a single entity; type queries, which target multiple entities of a single type; attribute queries, which target values of a particular entity attribute; and relation queries, which aim to find how two or more entities or types are related. Entity queries and type queries accounted for more than 50% of a query log sampled in their study, whereas attribute and relation queries accounted for just over 5%. Other works focused on more specific intents, such as a question intent [40], which targets answers to the question expressed in the query. In our experiments, we use an intent taxonomy comprising the three major classes described in these studies, namely, entity, type, and question queries, as well as an additional class including less represented intents, such as attribute and relation queries.

In addition to detecting query intents, several approaches have attempted to adapt the ranking produced for a query in light of some identified query property, such as its intent. For instance, Yom-Tov et al. [42] proposed to adaptively expand a query depending on its predicted difficulty. Kang and Kim [23] proposed to apply

[3] https://www.w3.org/TR/rdf-sparql-query/

different hand-crafted ranking models for queries with a predicted informational, navigational, or transactional intent. However, such a hard intent classification may eventually harm the effectiveness of an adaptive approach, when queries of different intents benefit from a single ranking model [14]. To mitigate this effect, instance-based classification approaches have been used to identify similar queries (as opposed to queries with the same predicted intent) for training a ranking model. For example, Geng et al. [19] resorted to nearest neighbor classification for building training sets for a given test query. Relatedly, Peng et al. [30] proposed to estimate the benefit of multiple candidate ranking models for a given query by examining training queries that are affected by these models in a similar manner. In the context of search result diversification, Santos et al. proposed adaptive approaches for estimating the coverage of different query aspects given their predicted intent [38] as well as for estimating when to diversify given the predicted ambiguity of the query [36]. Our proposed approach resembles these adaptive ranking approaches as we also resort to query intent classification as a trigger for ranking adaptation. Nonetheless, to the best of our knowledge, our approach is the first attempt to produce adaptive learning to rank models for a semantic search task.

3 INTENT-AWARE RANKING ADAPTATION FOR SEMANTIC QUERY ANNOTATION

Annotating queries with semantic information is an important step towards an improved query understanding [1]. Given a query, our goal is to automatically annotate it with semantic resources mined from a knowledge base, including named entities, attributes, relations, etc. For instance, the query "*us president*" could be annotated with arguably relevant semantic resources including "Donald Trump", "Federal Government", "White House." In this paper, we hypothesize that the relevance of a semantic resource given a query depends on the intent underlying this query. For the previous example, knowing that the query "*us president*" targets information around a single entity could promote alternative semantic resources including "Inauguration", "First 100 days", and "Controversies".

In this section, we propose an intent-aware framework for learning to rank semantic query annotations. In particular, we posit that the probability $P(r|q)$ that a given semantic resource r satisfies the user's query q should be estimated in light of the possible intents $i \in \mathcal{I}$ underlying this query. Formally, we define:

$$P(r|q) = \sum_{i \in \mathcal{I}} P(i|q)\, P(r|q, i), \tag{1}$$

where $P(i|q)$ is the probability that query q conveys an intent i, with $\sum_{i \in \mathcal{I}} P(i|q) = 1$, and $P(r|q, i)$ is the probability of observing semantic resource r given the query and this particular intent.

In Figure 1, we describe the three core components of our framework. In particular, the *query intent classification* and the *intent-specific learning to rank* components rely on supervised learning approaches to estimate $P(i|q)$ and $P(r|q, i)$, respectively, for each intent $i \in \mathcal{I}$. In turn, the *intent-aware ranking adaptation* component implements two alternative policies to suit the final ranking to the detected intents of each individual query.

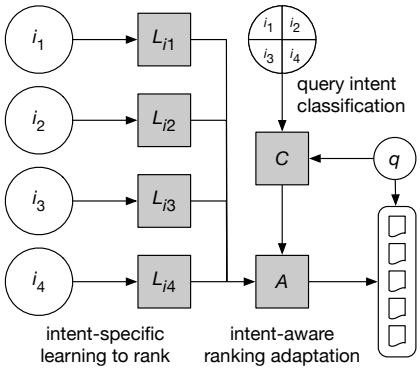

Figure 1: Intent-aware semantic query annotation. Each intent-specific ranking model L_i is learned on a query set comprising only queries with intent i. The query intent classification model C is learned on a set comprising queries of various intents. The intent-aware ranking adaptation strategy A uses the query intent classification outcome to decide on how to leverage the intent-specific ranking models.

3.1 Query Intent Classification

The first component of our framework is responsible for predicting the possible intents underlying a query [8]. For this task, we adopt a standard multi-class classification approach. In particular, we aim to learn a query classification model $C : \mathcal{X} \rightarrow \mathcal{Y}$ mapping the input space $\mathcal{X}$ into the output space $\mathcal{Y}$. Our input space $\mathcal{X}$ comprises m learning instances $\{\vec{x}_j\}_{j=1}^{m}$, where $\vec{x}_j = \Phi(q_j)$ is a feature vector representation of query q_j as produced by a feature extractor Φ. In turn, our output space $\mathcal{Y}$ comprises m labels $\{y_j\}_{j=1}^{m}$, where y_j corresponds to one of the target intents $i \in \mathcal{I}$ assigned to query q_j by a human annotator. To learn an effective classifier C, we experiment with several classification algorithms in Section 5.2.

Table 1 presents the features we use to represent a query for intent classification. We use a total of 31 simple features, including both lexical as well as semantic ones. Lexical features like number of query terms and mean query term size can help detect, for example, natural language queries, which are usually longer than others. In addition, part-of-speech tags can help identify question queries, indicating the presence of wh-pronouns (e.g., what, where, why, when). Lastly, semantic features include the number of categories and number of ontology classes returned when using the query to search a knowledge base. Our intuition is that queries seeking for a specific entity will probably return fewer categories or ontology classes than queries seeking for a list of entities. For instance, the query "*eiffel*" returns only 5 categories, while the query "*list of films from the surrealist category*" returns more than 103,000.

3.2 Intent-Specific Learning to Rank

The second component of our framework aims to produce multiple ranking models, each one optimized for a specific query intent $i \in \mathcal{I}$. To this end, we resort to learning to rank [24]. Analogously to our query intent classification models in Section 3.1, our goal is to learn an intent-specific ranking model $L_i : \mathcal{V} \rightarrow \mathcal{W}$ mapping the input space $\mathcal{V}$ into the output space $\mathcal{W}$. Our input space

Table 1: Query features for intent classification.

#	Feature	Qty
1	No. of query terms	1
2	Avg. query term size (in characters)	1
3	No. of matched categories in DBPedia	1
4	No. of matched ontology classes in DBPedia	1
5	No. of POS tags of different types	27
	TOTAL	31

includes n learning instances $\{\vec{V}_j\}_{j=1}^{n}$, where $\vec{V}_j = \Omega(q_j, \mathcal{R}_j)$ is a feature matrix representation (produced by some feature extractor Ω) of a sample of semantic resources $r \in \mathcal{R}_j$ retrieved for query q_j annotated with intent i. In our experiments, $\mathcal{R}_j$ is produced using BM25 [33], although any unsupervised ranking technique could have been used for this purpose. Our output space $\mathcal{W}$ comprises n label vectors $\{\vec{W}_j\}_{j=1}^{n}$, where $\vec{W}_j$ provides relevance labels for each semantic resource $r \in \mathcal{R}_j$. To learn an effective ranking model L_i for each intent $i \in \mathcal{I}$, we use LambdaMART [41], which represents the current state-of-the-art in learning to rank [13].

Table 2 lists all 216 features used to represent each semantic resource $r \in \mathcal{R}_j$. Features #1-#6 are content-based features commonly used in the learning to rank literature [24], such as number of tokens, BM25, coordination level matching (CLM), TF, and IDF scores. These are computed in a total of 8 descriptive fields of r, such as name, attributes, categories (see Section 4.1 for a full description). Since TF and IDF are defined on a term-level, query-level scores are computed using multiple summary statistics (sum, min, max, avg, var). Finally, CLM, TF, IDF, and TF-IDF are computed for both unigrams and bigrams. Next, features #7-#14 are semantic features derived from a knowledge base. For instance, feature #7 indicates whether r is an entity directly mentioned in the query, while feature #8 considers the number of direct connections between r and all entities mentioned in the query. As an example of the latter feature, in the query "*songs composed by michael jackson*", the candidate resource "Thriller" will be directly related to the entity "Michael Jackson" (present in the query). For both features, we use DBPedia Spotlight[4] for entity recognition in queries. Features #9-#14 are query-independent features quantifying the connectivity of each candidate resource r with respect to other resources in the knowledge base (e.g., entities, categories, ontology classes).

To keep our approach general, instead of handpicking features more likely to be useful for a particular intent, we use the same 216 available features when learning every intent-specific model L_i. To ensure that the learned model L_i is indeed optimized to its target intent i, intent-specific learning is achieved by using one training query set per intent, as illustrated in Figure 1.

3.3 Intent-Aware Ranking Adaptation

Sections 3.1 and 3.2 described supervised approaches for learning a query intent classification model C as well as multiple intent-specific ranking models L_i for all $i \in \mathcal{I}$. Importantly, all of these models are learned offline. When an unseen query q is submitted online, we must be able to return a ranking of semantic resources well suited to the target intent of q. Because we tackle query intent classification as a multi-class problem, we can actually estimate the probability $P(i|q)$ of different intents $i \in \mathcal{I}$ given the query q.

[4]http://spotlight.dbpedia.org/

Table 2: Semantic resource features for learning to rank. Features marked as 'Bi' are computed also for bigrams.

#	Feature	Bi	Qty
1	No. of tokens (per-field)		8
2	BM25 (per-field)		8
3	CLM (per-field)	✔	16
4	TF (per-field sum, min, max, avg, var)	✔	80
5	IDF (per-field sum)	✔	16
6	TF-IDF (per-field sum, min, max, avg, var)	✔	80
7	Matching entity		1
8	No. of direct relations with query entities		1
9	No. of matched relations with query terms		1
10	No. of inlinks		1
11	No. of outlinks		1
12	No. of linked ontology classes		1
13	No. of linked categories		1
14	No. of linked entities		1
	TOTAL		216

To exploit this possibility, we devise two strategies to adapt the ranking produced for a query q to the target intent(s) of this query. Our first strategy, called *intent-aware switching*, assigns each query a single intent, namely, the most likely one as predicted by the intent classification model C. For instance, for a target set of intents $\mathcal{I} = \{i_1, i_2, i_3\}$ of which i_1 is predicted as the most likely for q, we could instantiate Equation (1) with $P(i_1|q) = 1$, $P(i_2|q) = 0$, and $P(i_3|q) = 0$. As a result, only $P(r|q, i_1)$ (estimated via ranking model L_1) would have an impact on the final ranking, such that:

$$P(r|q) = P(r|q, i_1).$$

Some queries may have no clear winning intent. Other queries may prove simply difficult to classify correctly. To cope with uncertainty in intent classification, we propose a second ranking adaptation strategy, called *intent-aware mixing*. In this strategy, we use the full probability distribution over intents predicted by the classification model C to produce the final ranking for q. In the aforementioned example, suppose the predicted intent distribution is $P(i_1|q) = 0.7$, $P(i_2|q) = 0.2$, and $P(i_3|q) = 0.1$. Leveraging this distribution directly in Equation (1), we have a mixture of intent-specific ranking models contributing to the final ranking:

$$\begin{aligned} P(r|q) = {} & 0.7 \times P(r|q, i_1) \\ & + 0.2 \times P(r|q, i_2) \\ & + 0.1 \times P(r|q, i_3). \end{aligned}$$

To assess the effectiveness of our proposed intent-aware ranking adaptation strategies for semantic query annotation, in the next section, we compare these strategies to each other as well as to state-of-the-art intent-agnostic approaches from the literature.

4 EXPERIMENTAL SETUP

In this section, we detail the experimental setup that supports the evaluation of our proposed intent-aware semantic query annotation

approach introduced in Section 3. In particular, our experiments aim to answer the following research questions:

Q1. Do different intents benefit from different ranking models?
Q2. How accurately can we predict the intent of each query?
Q3. How effective is our semantic query annotation approach?
Q4. What queries are improved the most and the least?

In the following, we describe the knowledge base, queries, relevance judgments, and intent taxonomy used in our experiments. We also describe the baselines used for comparison and the procedure undertaken to train and test them as well as our own models.

4.1 Knowledge Base

The knowledge base used in our experiments is the English portion of DBPedia 3.7,[5] which comprises RDF triples extracted from Wikipedia dumps generated in late July 2011. This version of DBPedia contains information on more than 3.6 million entities organized in over 170,000 categories and 320 ontology classes in a 6-level deep hierarchy. We indexed this knowledge base using Elasticsearch 1.7.5[6] for textual content and Titan 0.5.4[7] for the underlying graph structure. RDF triples were parsed to created a fielded content representation. Following Zhiltsov et al. [44], we indexed the *Names*, *Attributes*, *Categories*, *Similar entity names* and *Related entity names* fields. In addition, we included three other fields: *Ontology classes*, *URL* and a special field *All*, concatenating the available content from all fields. Indexed terms were lower-cased, stemmed using Krovetz stemmer and standard stopwords were removed.

4.2 Queries, Relevance Judgments, and Intents

We use a publicly available benchmark[8] built on top of DBPedia 3.7, which comprises a total of 485 queries from past semantic search evaluation campaigns [3]. In total, there are 13,090 positive relevance judgments available. While some of these include graded labels, for a homogeneous treatment of all queries, we consider relevance as binary. The benchmark covers a wide variety of query intents, including entity, type, relation and attribute queries, as well as queries with a question intent. Following past research [3, 28, 44], we organize these queries into four intent-specific query sets, the salient statistics of which are described in Table 3:

- E: entity queries (e.g., "*orlando florida*");
- T: type queries (e.g., "*continents in the world*");
- Q: question queries (e.g., "*who created wikipedia?*");
- O: queries with other intents, including less represented ones, such as relation queries and attribute queries.

4.3 Retrieval Baselines

We compare our approach to multiple intent-agnostic baselines from the literature. As a vanilla ad-hoc search baseline, we consider BM25 with standard parameter settings ($k_1 = 1.2$, $b = 0.8$). To assess the effectiveness of our intent-aware ranking adaptation strategies introduced in Section 3.3, we further contrast them to two intent-agnostic strategies, which consistently apply a single ranking model for all queries, regardless of their target intent. As illustrated in Table 4, the *fixed* strategy applies a model L_i learned on one intent-specific query set, whereas the *oblivious* strategy applies a model L_R learned on a set of random queries. For a fair comparison, both of these baseline strategies as well as our own intent-aware switching and mixing strategies use the same learning algorithm (LambdaMART) and ranking features (all 216 features in Table 2). Lastly, we further contrast our approach to FSDM [44] (see Section 2.1) as a representative of the current state-of-the-art.

Table 3: Statistics of the intent-specific query sets used in our evaluation. Length and qrels denote per-query averages of query length and positive judgements in each set.

Set	Campaign [3]	Queries	Length	Qrels
E	SemSearch ES	130	2.7	8.7
T	INEX-XER, SemSearch LS, TREC Entity	115	5.8	18.4
Q	QALD-2	140	7.9	41.5
O	INEX-LD	100	4.8	37.6
TOTAL		485	5.3	26.55

4.4 Training and Test Procedure

For a fair comparison between our intent-aware semantic query annotation approach and the intent-agnostic baselines described in Section 4.3, we randomly downsample all query sets in Table 3 until they reach 100 queries each (i.e., the number of queries in the smallest query set, namely, O). This procedure ensures the learning process is not biased towards any particular intent. To learn an intent-specific model L_i for each intent $i \in \mathcal{I} = \{E, T, Q, O\}$, we perform a 5-fold cross validation in the corresponding query set from Table 3. For the oblivious strategy, the intent-agnostic model L_R is also learned via 5-fold cross validation on a set of 100 queries sampled uniformly at random from the four intent-specific query sets after downsampling. This multi-intent query set is also used to tune the parameters of FSDM [44] for different concepts (unigrams, ordered, an unordered bigrams) and each of the fields listed in Section 4.1. In each cross-validation round, we use three partitions (60 queries) for training, one partition (20 queries) for validation, and one partition (20 queries) for testing.

Learning to rank is performed using the LambdaMART implementation in RankLib 2.7,[9] optimizing for normalized discounted cumulative gain at the top 100 results (nDCG@100). LambdaMART is deployed with default hyperparameter settings,[10] with 1,000 trees with 10 leaves each, minimum leaf support 1, unlimited threshold candidates for tree splitting, learning rate 0.1, and early stopping after 100 non-improving iterations. All results are reported as averages of all test queries across the five cross-validation rounds. In particular, we report nDCG@10, precision at 10 (P@10), and mean average precision (MAP). All evaluation metrics are calculated on the top 100 results returned by each approach. To check for statistically significant differences among them, we use a two-tailed paired

[5] http://wiki.dbpedia.org/data-set-37
[6] https://www.elastic.co/products/elasticsearch
[7] http://titan.thinkaurelius.com
[8] http://bit.ly/dbpedia-entity
[9] https://sourceforge.net/p/lemur/wiki/RankLib%20How%20to%20use/
[10] Hyperparameter tuning on validation data showed no significant improvements in our preliminary tests.

Table 4: Example application of intent-agnostic (baseline) and intent-aware ranking adaptation strategies.

	intent-agnostic					intent-aware	
i	fixed-E	fixed-T	fixed-Q	fixed-O	oblivious	switching	mixing
E	L_E	L_T	L_Q	L_O	L_R	L_E	$\sum_i w_i L_i$
T	L_E	L_T	L_Q	L_O	L_R	L_T	$\sum_i w_i L_i$
Q	L_E	L_T	L_Q	L_O	L_R	L_Q	$\sum_i w_i L_i$
O	L_E	L_T	L_Q	L_O	L_R	L_O	$\sum_i w_i L_i$

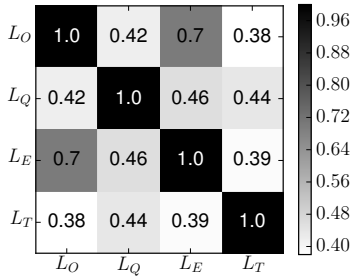

Figure 2: Spearman's correlation coefficient for feature importance across pairs of intent-specific ranking models.

t-test and write △ (▽) and ▲ (▼) to denote significant increases (decreases) at the 0.05 and 0.01 levels, respectively. A further symbol ∘ is used to denote no significant difference.

5 EXPERIMENTAL EVALUATION

In this section, we empirically evaluate our approach in order to answer the four research questions stated in Section 4. In the following, we address each of these questions in turn.

5.1 Intent Specificity

The core hypothesis of our proposal is that different queries may benefit from a ranking model optimized to their intent. To verify this hypothesis, we address *Q1*, by assessing the specificity of ranking models optimized to the four intents described in Table 3. To this end, Figure 2 correlates the importance assigned to all 216 features by each intent-specific ranking model L_i, for $i \in \mathcal{I} = \{E, T, Q, O\}$. Feature importance is quantified using the least square improvement criterion proposed by Lucchese et al. [25] for gradient boosted regression tree learners, such as LambdaMART. From Figure 2, we observe a generally low correlation ($\rho < 0.5$) between models, except for the L_E and L_O models, with $\rho \approx 0.7$.

Table 5 lists the five most important features for each intent-specific model. The entity-oriented L_E model gives importance to features related to the occurrence of bigrams in the name and similar entities fields. For instance, the query "*martin luther king*" expects semantic resources named "Martin Luther King III" and "Martin Luther King High School." While the type-oriented L_T model considers a variety of distinct features, two features related to the categories field are present in the top 5, which are useful for queries like "*state capitals of the united states of america*." The question-oriented L_Q model gives importance to features describing the relation between entities and ontology classes, derived from both content fields as well as the graph structure underlying the knowledge base. These can help identify relevant resources linked to an entity in the query through qualified relations, as in the query "*who was the successor of john f. kennedy?*" Lastly, the L_O model, which is optimized on a set comprising queries of various intents, strongly favors content-based features, which are arguably effective for broad queries such as "*einstein relativity theory.*" Recalling question *Q1*, these results provide a strong indication of the specificity of different models to queries of different intents.

Table 5: Top 5 features per ranking model.

	#	Feature
L_E	1	TF-IDF sum of bigrams in *similar entities*
	2	Matching entity
	3	TF sum of bigrams in *similar entities*
	4	TF avg of bigrams in *similar entities*
	5	TF-IDF max of bigrams in *similar entities*
L_T	1	CLM in *categories*
	2	CLM in *all content*
	3	No. of inlinks
	4	No. of tokens in *similar entities*
	5	TF-IDF sum of bigrams in *categories*
L_Q	1	BM25 in *ontology classes*
	2	No. of matched relations with query terms
	3	No. of direct relations with query entities
	4	No. of inlinks
	5	TF-IDF max of unigrams in *ontology classes*
L_O	1	TF sum of bigrams in *name*
	2	BM25 in *name*
	3	TF-IDF max of unigrams in *categories*
	4	TF-IDF max of bigrams in *name*
	5	TF-IDF var of bigrams in *all content*

5.2 Intent Classification Accuracy

The results in the previous experiment suggest that exploiting the specificity of different query intents may result in more effective ranking models. Before investigating whether this is indeed the case, in this section, we address *Q2*, with the aim of establishing what level of query intent detection accuracy can be attained in practice. To this end, we experiment with a range of traditional classification algorithms implemented in Scikit-learn 0.17.1,[11] optimized via 5-fold cross validation using the same partitions leveraged for learning to rank, as detailed in Section 4.4. Table 6 reports

[11] http://scikit-learn.org/

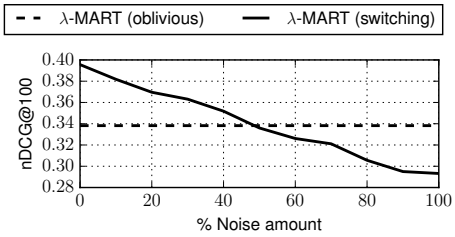

Figure 3: Semantic query annotation robustness for simulated intent classifiers of a range of accuracy levels.

intent classification accuracy averaged across test queries in all cross-validation rounds. As shown in the table, the most accurate intent classifier is learned via stochastic gradient descent with a log loss, which performs an incremental logistic regression. As a result, we will use this classifier in the remainder of our experiments. Other effective classification algorithms include random forest and bagging, while AdaBoost has the lowest accuracy.

Table 6: Query intent classification accuracy.

Algorithm	Accuracy
AdaBoost	67.00%
Support Vector Machines	74.00%
Gradient Boosting	75.75%
Bagging	76.00%
Random Forest	76.50%
Logistic Regression	**77.00%**

The top performing classifier in Table 6 still leaves room for further improvement in intent classification accuracy. An interesting question here is whether this level of accuracy is enough for an effective deployment of our proposed intent-aware semantic query annotation approach. To further investigate the role of the intent classification component in our approach, we measure the impact of a range of simulated intent classifiers on the effectiveness of the produced ranking of semantic annotations. In particular, starting from a perfect intent classifier (i.e., an oracle), we gradually introduce noise in the classification outcome by replacing the correct intent with a random one, up to the point where the classification itself becomes a random guess of the four available intents (i.e., *E*, *T*, *Q*, and *O*). As shown in Figure 3, our intent-aware switching strategy can outperform the intent-agnostic oblivious strategy with up to 50% of random noise in intent classification, which is a remarkable result. Recalling *Q2*, the experiments in this section demonstrate that accurate intent classification is feasible, and that the overall ranking annotation performance is robust to a considerable amount of noise in the predicted intents.

5.3 Annotation Effectiveness

Section 5.1 showed the promise of leveraging intent-specific ranking models, while Section 5.2 demonstrated that achieving this promise is feasible with reasonably accurate query intent classifiers. In this section, we address *Q3*, by assessing the effectiveness of our intent-aware semantic query annotation approach in contrast to the various baselines described in Section 4.3. These include BM25 as a vanilla ad-hoc search baseline, FSDM as a representative of the current state-of-the-art, and multiple deployments of LambaMART using baseline intent-agnostic ranking adaptation strategies (fixed and oblivious) as well as our proposed intent-aware strategies (switching and mixing). Table 7 summarizes the results of this investigation in terms of P@10, nDCG@10, and MAP averaged across all 400 test queries from the four query sets in Table 3.[12] In each row describing baseline results (the top half of the table), a first of the symbols introduced in Section 4.4 denotes a statistically significant difference (or lack thereof) with respect to LambdaMART (switching), whereas a second symbol denotes potential differences with respect to LambdaMART (mixing). A further symbol is shown alongside LambdaMART (switching) to denote a significant difference (or lack thereof) with respect to LambdaMART (mixing).

Table 7: Comparison of intent-agnostic (BM25, FSDM, LambdaMART fixed and oblivious) and intent-aware (LambdaMART switching and mixing) semantic query annotation.

	P@10	nDCG@10	MAP
BM25	$0.181^{\blacktriangledown\blacktriangledown}$	$0.250^{\blacktriangledown\blacktriangledown}$	$0.163^{\blacktriangledown\blacktriangledown}$
FSDM	$0.204^{\blacktriangledown\blacktriangledown}$	$0.289^{\blacktriangledown\blacktriangledown}$	$0.195^{\blacktriangledown\blacktriangledown}$
LambdaMART			
(fixed-E)	$0.178^{\blacktriangledown\blacktriangledown}$	$0.244^{\blacktriangledown\blacktriangledown}$	$0.162^{\blacktriangledown\blacktriangledown}$
(fixed-T)	$0.202^{\blacktriangledown\blacktriangledown}$	$0.275^{\blacktriangledown\blacktriangledown}$	$0.172^{\blacktriangledown\blacktriangledown}$
(fixed-Q)	$0.152^{\blacktriangledown\blacktriangledown}$	$0.215^{\blacktriangledown\blacktriangledown}$	$0.139^{\blacktriangledown\blacktriangledown}$
(fixed-O)	$0.188^{\blacktriangledown\blacktriangledown}$	$0.260^{\blacktriangledown\blacktriangledown}$	$0.163^{\blacktriangledown\blacktriangledown}$
(oblivious)	$0.192^{\blacktriangledown\blacktriangledown}$	$0.276^{\blacktriangledown\blacktriangledown}$	$0.178^{\blacktriangledown\blacktriangledown}$
(switching)	$0.227^{\blacktriangledown}$	$0.329^{\blacktriangledown}$	$0.219^{\blacktriangledown}$
(mixing)	**0.243**	**0.346**	**0.229**

From Table 7, we first observe that FSDM performs strongly, outperforming all intent-agnostic variants deployed with LambdaMART, which confirms its effectiveness as a representative of the state-of-the-art. Also of note is the fact that a single model trained on a set of multiple intents using the oblivious strategy cannot consistently improve upon the best performing intent-specific model, produced by the fixed-T strategy. In contrast, both of our intent-aware ranking adaptation strategies are able to consistently leverage the best characteristics of each individual intent, significantly outperforming all intent-agnostic baselines in all settings. In particular, compared to FSDM, our switching strategy improves by up to 11% in P@10, 14% in nDCG@10, and 12% in MAP. Compared to the best performing intent-agnostic strategy under LambdaMART (fixed-T), gains are as high as 12% in P@10, 20% in nDCG@10, and 27% in MAP. Lastly, we also note that our mixing strategy further significantly improves upon the switching strategy. This result suggests that merging multiple intent-specific models (the mixing strategy) can be safer than applying a single model associated with the most likely query intent (the switching strategy). Recalling *Q3*, these results attest the effectiveness of our intent-aware ranking adaptation for semantic query annotation.

[12] Effectiveness breakdown analyses per query intent and various other query characteristics are presented in Section 5.4.

5.4 Breakdown Analyses

The previous analysis demonstrated the effectiveness of our approach on the entire set of 400 queries. To further shed light on the reasons behind such an effective performance, we address question *Q4*, by analyzing the improvements brought by our approach for queries with different intents, lengths, and difficulty.

5.4.1 Analysis by Query Intent. Table 8 breaks down the results in Table 7 according to the target intent of each query. For brevity, only the best among the fixed strategy variants is shown. Note that while our approach aims to predict the correct intent of each query, there is no guarantee that a perfect intent classification will be achieved, as discussed in Section 5.2. Hence, it is important to understand how well our approach performs on queries of each target intent. From Table 7, as expected, the best fixed strategy for each group of queries is that optimized for the group itself (e.g., fixed-E is the best fixed strategy for entity queries—the *E* group). Nonetheless, our intent-aware mixing strategy is the most consistent across all groups, with effectiveness on a par with the best fixed strategy for each group. Compared to our switching strategy, the mixing strategy is particularly effective for type queries (the *T* group), with statistical ties for all other groups. Regarding performance differences across the target intents, we note that all approaches achieve their best absolute performance on *E* queries followed by queries with other intents (the *O* group), which also includes entity queries. The effective results attained even by the simple BM25 baseline suggest that queries with these intents are well handled by content-based approaches.

Compared to the intent-agnostic FSDM baseline, our largest improvements are observed for type queries (the *T* group) and question queries (the *Q* group). For *T* queries, the structure-based features exploited by our learning to rank approach bring only small improvements, as observed by contrasting LambdaMART (oblivious) with FSDM. However, with our proposed intent-aware ranking adaptation strategies, further marked improvements are observed, with the mixing strategy significantly improving upon the oblivious strategy by up to 25% in P@10, 35% in nDCG@10, and 44% in MAP. For *Q* queries, both the extra features exploited via learning to rank as well as our ranking adaptation strategies help, with the switching strategy improving even further compared to the oblivious one by up to 56% in P@10, 51% in nDCG@10, and 50% in MAP. Figure 4 further illustrates the consistent improvements in terms of nDCG@100 attained by our intent-aware strategies (here represented by the mixing strategy) compared to the intent-agnostic oblivious baseline. Indeed, not only does mixing improve more queries than it hurts compared to oblivious, but it also shows larger increases and smaller decreases throughout queries of all four intents. Analyzing each intent separately, the most noticeable difference can be observed for *Q* queries, with mixing performing better for 50% of the queries and losing in only 10%. For *E* and *T* queries, the differences in nDCG are not as high, but mixing is still superior for 60% of the queries. The smallest gap between the two strategies appears in *O* queries, although once again mixing performs better for 60% the queries.

5.4.2 Analysis by Query Length. Continuing our detailed analysis, Table 9 breaks down the results from Table 7 according to the length of each query. In particular, we consider three groups of queries: short queries, with 1 or 2 terms (74 queries); medium queries, with 3 or 4 terms (193 queries); and long queries, with 5 or more terms (133 queries). From Table 9, we observe relatively higher performances of all approaches on short queries compared to those of other lengths. FSDM delivers a particularly strong performance on this group, with only a small gap from our mixing strategy, which is the overall best. This can be explained by FSDM's previously discussed effectiveness on *E* queries, which have only 2.7 terms on average. Compared to the oblivious strategy, mixing brings substantial and significant improvements, once again demonstrating the benefits of an intent-aware ranking adaptation. For medium and long queries (5 or more terms), both of our intent-aware strategies bring even more pronounced improvements compared to all intent-agnostic baselines, with the top performing mixing strategy outperforming the oblivious strategy by up to 32% in P@10, 30% in nDCG@10, and 36% in MAP. This tendency is somewhat expected given the effective performance observed

Table 8: Effectiveness breakdown by query intent.

	P@10	nDCG@10	MAP
	***E* queries (100 queries)**		
BM25	$0.240^{\blacktriangledown\blacktriangledown}$	$0.416^{\blacktriangledown\blacktriangledown}$	$0.320^{\blacktriangledown\blacktriangledown}$
FSDM	$0.286^{\circ\circ}$	$0.499^{\circ\circ}$	$0.396^{\circ\circ}$
LambdaMART			
(fixed-E)	$0.293^{\triangle\circ}$	$0.498^{\circ\circ}$	$0.387^{\circ\circ}$
(oblivious)	$0.239^{\blacktriangledown\blacktriangledown}$	$0.434^{\blacktriangledown\blacktriangledown}$	$0.329^{\blacktriangledown\blacktriangledown}$
(switching)	0.282°	0.486°	0.377°
(mixing)	**0.297**	**0.502**	**0.390**
	***T* queries (100 queries)**		
BM25	$0.190^{\triangledown\blacktriangledown}$	$0.193^{\blacktriangledown\blacktriangledown}$	$0.141^{\blacktriangledown\blacktriangledown}$
FSDM	$0.211^{\circ\blacktriangledown}$	$0.223^{\circ\blacktriangledown}$	$0.167^{\circ\blacktriangledown}$
LambdaMART			
(fixed-T)	$\mathbf{0.289}^{\blacktriangle\circ}$	$\mathbf{0.327}^{\blacktriangle\circ}$	$\mathbf{0.219}^{\blacktriangle\circ}$
(oblivious)	$0.216^{\circ\blacktriangledown}$	$0.225^{\circ\blacktriangledown}$	$0.146^{\blacktriangledown\blacktriangledown}$
(switching)	$0.232^{\blacktriangledown}$	$0.260^{\blacktriangledown}$	$0.185^{\blacktriangledown}$
(mixing)	0.271	0.303	0.210
	***Q* queries (100 queries)**		
BM25	$0.060^{\blacktriangledown\blacktriangledown}$	$0.108^{\blacktriangledown\blacktriangledown}$	$0.077^{\blacktriangledown\blacktriangledown}$
FSDM	$0.061^{\blacktriangledown\blacktriangledown}$	$0.127^{\blacktriangledown\blacktriangledown}$	$0.098^{\blacktriangledown\blacktriangledown}$
LambdaMART			
(fixed-Q)	$\mathbf{0.143}^{\circ\circ}$	$\mathbf{0.273}^{\circ\circ}$	$\mathbf{0.203}^{\circ\circ}$
(oblivious)	$0.091^{\blacktriangledown\blacktriangledown}$	$0.177^{\blacktriangledown\blacktriangledown}$	$0.132^{\blacktriangledown\blacktriangledown}$
(switching)	0.142°	0.267°	0.198°
(mixing)	0.141	0.266	0.194
	***O* queries (100 queries)**		
BM25	$0.235^{\circ\blacktriangledown}$	$0.282^{\circ\triangledown}$	$0.113^{\circ\circ}$
FSDM	$0.258^{\circ\circ}$	$0.308^{\circ\circ}$	$0.119^{\circ\circ}$
LambdaMART			
(fixed-O)	$0.259^{\circ\circ}$	$0.304^{\circ\circ}$	$0.113^{\circ\triangledown}$
(oblivious)	$0.221^{\blacktriangledown\blacktriangledown}$	$0.267^{\triangledown\blacktriangledown}$	$0.105^{\circ\blacktriangledown}$
(switching)	0.254°	0.305°	0.116°
(mixing)	**0.264**	**0.312**	**0.123**

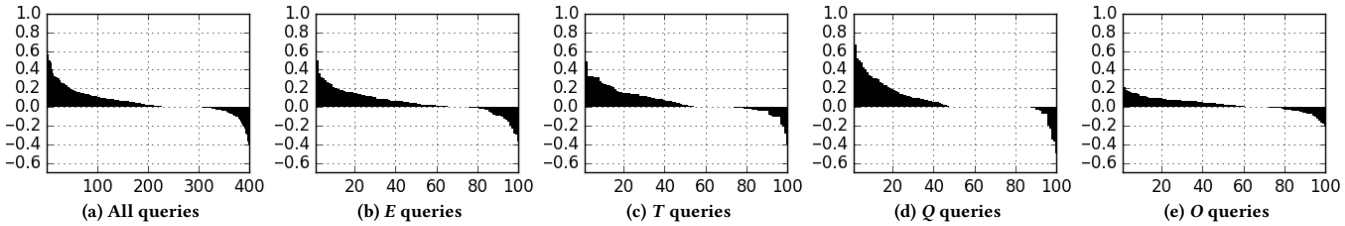

Figure 4: Differences in nDCG@100 between LambdaMART (mixing) and LambdaMART (oblivious) across: (a) all queries; (b) *E* queries; (c) *T* queries; (d) *Q* queries; (e) *O* queries. Positive values indicate mixing is better.

in Table 9 for the proposed intent-aware strategies on *Q* queries, which are typically longer (8 terms on average).

Table 9: Effectiveness breakdown by query length.

	P@10	nDCG@10	MAP
1 or 2 terms (74 queries)			
BM25	0.253▼▼	0.363▼▼	0.268▼▼
FSDM	0.323°°	0.456°°	0.331°°
LambdaMART			
(fixed-E)	0.324°°	0.453°°	0.327°°
(oblivious)	0.276▼▼	0.394▼▼	0.278▼▼
(switching)	0.324°	0.455°	0.324°
(mixing)	**0.337**	**0.465**	**0.332**
3 or 4 terms (193 queries)			
BM25	0.196▼▼	0.264▼▼	0.161▼▼
FSDM	0.221°▼	0.308▽▼	0.198▽▼
LambdaMART			
(fixed-T)	0.234°▼	0.307▽▼	0.182▼▼
(oblivious)	0.208▼▼	0.296▼▼	0.183▼▼
(switching)	0.240▼	0.349▼	0.227°
(mixing)	**0.265**	**0.376**	**0.239**
5 or more terms (133 queries)			
BM25	0.119▼▼	0.167▼▼	0.106▼▼
FSDM	0.114▼▼	0.171▼▼	0.115▽▼
LambdaMART			
(fixed-T)	0.141°▼	0.206°▽	0.134°▽
(oblivious)	0.121▼▼	0.181▼▼	0.116▼▼
(switching)	0.156°	0.230°	0.150▽
(mixing)	**0.160**	**0.236**	**0.158**

5.4.3 Analysis by Query Difficulty. To complete our breakdown analysis, we regroup all 400 queries in our investigation according to their difficulty. In particular, we consider three groups: difficult queries, with 3 or less relevant results in the ground-truth (108 queries); moderate queries, with 4 to 20 relevant results (184 queries); and easy queries, with more than 20 relevant results (108 queries). The results of this investigation are shown in Table 10. From the table, we note as expected that difficult queries generally incur in reduced precision at early ranks (as measured by both P@10 and nDCG@10), while easy queries tend to penalize recall at lower ranks (as measured by MAP). Nevertheless, our intent-aware adaptation strategies are once again the most effective across all groups of queries, with the mixing strategy consistently providing the overall best results. For difficult queries (3 or less relevant results), compared to the oblivious strategy, mixing improves by up to 19% in P@10, 24% in nDCG@10, and 26% in MAP. For easy queries (21 or more relevant results), improvements are as high as 24% in P@10, 27% in nDCG@10, and 39% in MAP.

Table 10: Effectiveness breakdown by query difficulty.

	P@10	nDCG@10	MAP
Difficult: 3 or less relevant results (108 queries)			
BM25	0.059°▼	0.224▼▼	0.179▼▼
FSDM	0.064°°	0.256▽▼	0.214°▽
LambdaMART			
(fixed-T)	0.062°▽	0.230▽▼	0.180▼▼
(oblivious)	0.063°▽	0.259▽▼	0.213▽▼
(switching)	0.069°	0.308°	0.260°
(mixing)	**0.075**	**0.322**	**0.268**
Moderate: 4 to 20 relevant results (184 queries)			
BM25	0.210▼▼	0.260▼▼	0.194▼▼
FSDM	0.245°▼	0.308°▼	0.235°▽
LambdaMART			
(fixed-O)	0.218▼▼	0.274▼▼	0.195▼▼
(oblivious)	0.214▼▼	0.276▼▼	0.202▼▼
(switching)	0.261▼	0.329▼	0.245▼
(mixing)	**0.279**	**0.345**	**0.257**
Easy: 21 or more relevant results (108 queries)			
BM25	0.255▼▼	0.259▼▼	0.094▼▼
FSDM	0.275▽▼	0.292▼▼	0.107▼▼
LambdaMART			
(fixed-T)	0.328°°	0.338°°	0.125°▼
(oblivious)	0.283▼▼	0.292▼▼	0.103▼▼
(switching)	0.328▽	0.352▽	0.134▼
(mixing)	**0.350**	**0.371**	**0.143**

Recalling *Q4*, the results in this section demonstrate the consistency of our intent-aware ranking adaptation strategies for semantic query annotation. Overall, both the switching and the mixing strategies achieve generally improved results for queries

of different target intents, lengths, and difficulty levels, often significantly. Particularly, question-oriented queries (the *Q* intent), long queries (queries with 5 or more terms), and moderate to easy queries (queries with 4 or more relevant results) are the ones that benefit the most from our intent-aware approach.

6 CONCLUSIONS

We presented a framework for learning to rank semantic annotations suitable to the intent of each individual query. Our approach predicts the intent of a target query and adapts the ranking produced for this query using one of two strategies: *switching*, which applies a ranking model trained on queries of the same intent as predicted for the target query, or *mixing*, which combines the results of multiple intent-specific ranking models according to their predicted likelihood for the target query. Extensive experiments on a publicly available benchmark demonstrated the effectiveness of our approach for semantic query annotation, with significant improvements compared to state-of-the-art intent-agnostic approaches. The results also attested the consistency of the observed improvements for queries of different intents, lengths, and difficulty levels.

In the future, we plan to assess the impact of intent-aware learning on frameworks other than learning to rank. Preliminary results in this direction show that the FSDM baseline, which is based on the Markov random fields framework, can be improved with an intent-aware approach to hyperparameter tuning, although with less marked gains compared to the ones observed in our experiments with feature-based models using learning to rank. Another direction for future investigation includes evaluating our approach with a larger intent taxonomy, including more queries with less common intents such as attribute and relation queries.

ACKNOWLEDGMENTS

This work was partially funded by projects InWeb (MCT/ CNPq 573871/2008-6) and MASWeb (FAPEMIG/PRONEX APQ-01400-14), and by the authors' individual grants from CNPq and FAPEMIG.

REFERENCES

[1] Omar Alonso and Hugo Zaragoza. 2008. Exploiting semantic annotations in information retrieval: ESAIR '08. *SIGIR Forum* 42, 1 (2008), 55–58.
[2] Kriztian Balog, Arien P. de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. 2009. Overview of the TREC 2009 Entity track. In *TREC*.
[3] Krisztian Balog and Robert Neumayer. 2013. A Test Collection for Entity Search in DBpedia. In *Proc. of SIGIR*. 737–740.
[4] Hannah Bast, Björn Buchhold, Elmar Haussmann, and others. 2016. Semantic Search on Text and Knowledge Bases. *Foundations and Trends in Information Retrieval* 10, 2-3 (2016), 119–271.
[5] Bin Bi, Hao Ma, Bo-June (Paul) Hsu, Wei Chu, Kuansan Wang, and Junghoo Cho. 2015. Learning to Recommend Related Entities to Search Users. In *Proc. of WSDM*. 139–148.
[6] Roi Blanco, Peter Mika, and Sebastian Vigna. 2011. Effective and Efficient Entity Search in RDF Data. In *Proc. of ISWC*. 83–97.
[7] Roi Blanco, Peter Mika, and Hugo Zaragoza. 2010. Entity Search Track Submission by Yahoo! Research Barcelona. In *Proc. of Entity Search Track*.
[8] David J. Brenes, Daniel Gayo-Avello, and Kilian Pérez-González. 2009. Survey and Evaluation of Query Intent Detection Methods. In *Proc. of WSCD*. 1–7.
[9] Andrei Broder. 2002. A Taxonomy of Web Search. *SIGIR Forum* 36, 2 (2002), 3–10.
[10] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2010. Ranking related entities: components and analyses. In *Proc. of CIKM*. 1079–1088.
[11] Marc Bron, Krisztian Balog, and Maarten De Rijke. 2013. Example Based Entity Search in the Web of Data. In *Proc. of ECIR*. 392–403.
[12] Stéphane Campinas, Renaud Delbru, Nur Aini Rakhmawati, Diego Ceccarelli, and Giovanni Tummarello. 2011. Sindice BM25F at SemSearch 2011. In *Proc. of SemSearch*.
[13] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview.. In *Yahoo! Learning to Rank Challenge*. 1–24.
[14] Nick Craswell and David Hawking. 2004. Overview of the TREC 2004 Web track. In *Proc. of TREC*.
[15] Arjen P. de Vries, Anne-Marie Vercoustre, James A. Thom, Nick Craswell, and Mounia Lalmas. 2007. Overview of the INEX 2007 Entity Ranking track. In *Proc. of INEX*. 245–251.
[16] Shady Elbassuoni and Roi Blanco. 2011. Keyword Search over RDF Graphs. In *Proc. of CIKM*. 237–242.
[17] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. 2009. Language-Model-Based Ranking for Queries on RDF-Graphs. In *Proc. of CIKM*. 977–986.
[18] Besnik Fetahu, Ujwal Gadiraju, and Stefan Dietze. 2015. Improving entity retrieval on structured data. In *Proc. of ISWC*. 474–491.
[19] Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. 2008. Query dependent ranking using k-nearest neighbor. In *Proc. of SIGIR*. 115–122.
[20] Ido Guy. 2016. Searching by Talking: Analysis of Voice Queries on Mobile Web Search. In *Proc. of SIGIR*. 35–44.
[21] Daniel M Herzig, Peter Mika, Roi Blanco, and Thanh Tran. 2013. Federated Entity Search Using On-the-fly Consolidation. In *Proc. of ISWC*. 167–183.
[22] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. 2000. Real Life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web. *Information Processing and Management* 36, 2 (2000), 207–227.
[23] In-Ho Kang and GilChang Kim. 2003. Query type classification for Web document retrieval. In *Proc. of SIGIR*. 64–71.
[24] Tie-Yan Liu and others. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.
[25] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. 2015. Speeding Up Document Ranking with Rank-based Features. In *Proc. of SIGIR*. 895–898.
[26] Donald Metzler and W. Bruce Croft. 2005. A Markov Random Field Model for Term Dependencies. In *Proc. of SIGIR*. 472–479.
[27] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. 2012. On the Modeling of Entities for Ad-Hoc Entity Search in the Web of Data. In *Proc. of ECIR*. 133–145.
[28] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. 2016. Parameterized Fielded Term Dependence Models for Ad-hoc Entity Retrieval from Knowledge Graph. In *Proc. of SIGIR*. 435–444.
[29] Paul Ogilvie and Jamie Callan. 2003. Combining Document Representations for Known-item Search. In *Proc. of SIGIR*. 143–150.
[30] Jie Peng, Craig Macdonald, and Iadh Ounis. 2010. Learning to select a ranking function. In *Proc. of ECIR*. 114–126.
[31] José R Pérez-Agüera, Javier Arroyo, Jane Greenberg, Joaquin Perez Iglesias, and Victor Fresno. 2010. Using BM25F for semantic search. In *Proc. of SemSearch*. 2.
[32] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc Object Retrieval in the Web of Data. In *Proc. of WWW*. 771–780.
[33] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. 1995. Okapi at TREC-4. In *Proc. of TREC*.
[34] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. 2004. A Hybrid Approach for Searching in the Semantic Web. In *Proc. of WWW*. 374–383.
[35] Daniel E. Rose and Danny Levinson. 2004. Understanding User Goals in Web Search. In *Proc. of WWW*. 13–19.
[36] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2010. Selectively diversifying web search results. In *Proc. of CIKM*. 1179–1188.
[37] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2010. Voting for related entities. In *Proc. of RIAO*. 1–8.
[38] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2011. Intent-aware Search Result Diversification. In *Proc. of SIGIR*. 595–604.
[39] Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. 2012. Combining Inverted Indices and Structured Search for Ad-hoc Object Retrieval. In *Proc. of SIGIR*. 125–134.
[40] Gilad Tsur, Yuval Pinter, Idan Szpektor, and David Carmel. 2016. Identifying Web Queries with Question Intent. In *Proc. of WWW*. 783–793.
[41] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. 2008. *Ranking, boosting, and model adaptation.* Technical Report. Technical report, Microsoft Research.
[42] Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. 2005. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proc. of SIGIR*. 512–519.
[43] Nikita Zhiltsov and Eugene Agichtein. 2013. Improving Entity Search over Linked Data by Modeling Latent Semantics. In *Proc. of CIKM*. 1253–1256.
[44] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *Proc. of SIGIR*. 253–262.