Traversing Semantically Annotated Queries for Task-oriented Query Recommendation

Arthur Câmara
TU Delft
Delft, The Netherlands
a.barbosacamara@tudelft.nl

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

ABSTRACT

As search systems gradually turn into intelligent personal assistants, users increasingly resort to a search engine to accomplish a complex task, such as planning a trip, renting an apartment, or investing in stocks. A key challenge for the search engine is to understand the user's underlying task given a sample query like "tickets to panama", "studios in los angeles", or "spotify stocks", and to suggest other queries to help the user complete the task. In this paper, we investigate several strategies for query recommendation by traversing a semantically annotated query log using a mixture of explicit and latent representations of entire queries and of query segments. Our results demonstrate the effectiveness of these strategies in terms of utility and diversity, as well as their complementarity, with significant improvements compared to state-of-the-art query recommendation baselines adapted for this task.

CCS CONCEPTS

• **Information systems → Query suggestion**.

KEYWORDS

Task understanding; query recommendations; query embeddings

ACM Reference Format:
Arthur Câmara and Rodrygo L. T. Santos. 2019. Traversing Semantically Annotated Queries for Task-oriented Query Recommendation. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3346994

1 INTRODUCTION

Given a query representing a user's complex task, which may encompass multiple subtasks, as seen in Figure 1, the goal of a task-oriented query recommendation system is to understand the underlying user task and generate a set of key phrases that are useful toward completing as many of these subtasks as possible [13]. This problem can be seen as a specialization of the query recommendation problem, which has been extensively investigated over the past decade. In particular, query recommendation approaches leverage a plethora of ranking signals from a query log in order to infer the relevance of a recommendation given an input query [1, 3, 12].

```
        Task: cure indigestion
 Freebase ID: /m/04kl78
    Subtasks: * home remedy for indigestion
              * food to prevent indigestion
              * medicines that cure indigestion
```

Figure 1: Excerpt of TREC 2016 Tasks track query #7.

Notwithstanding their maturity, current query recommendation approaches may fail to tackle the complex nature of task-oriented search [13]. As illustrated in Figure 1, task-oriented search requires producing recommendations that cover diverse subtasks centered around some named entity given a potentially rare or even unseen query. To address these challenges, we propose a two-stage approach. Firstly, to promote the understanding of tasks underlying rare or even unseen queries, we model semantically annotated query segments as a bipartite graph connecting named entities and the contexts where these entities appear in a query log. Secondly, to produce recommendations that cover a wide variety of subtasks, we investigate multiple strategies for traversing the entity-context graph given an input query. Through a comprehensive analysis using the experimentation paradigm provided by the TREC Tasks track [13] enriched via crowdsourcing, we show that these strategies are complementary and that their combination consistently outperforms state-of-the-art query recommendation baselines from the literature, particularly for long-tail as well as hard queries.

2 RELATED WORK

Among classical query recommendation approaches, Baeza-Yates and Tiberi [1] sought to promote queries frequently co-clicked with the input query. Relatedly, Boldi et al. [3] used a variety of syntax and time signals to build a machine learning model that connects queries with similar intents, producing recommendations via biased random walks on the resulting "query-flow" graph. Recently, approaches using deep neural networks have also been proposed, with major improvements over the previous state-of-the-art. For instance, Sordoni et al. [12] used a recurrent encoder-decoder network which captures the sequential nature of words in a query and of queries in a session for generating new recommendations.

3 ENTITY-CONTEXT GRAPH

A user task can target a variety of subjects, from planning a trip ("tickets to panama"), to building an investment portfolio ("spotify stocks"), to learning something new ("python tutorial"), with each task encompassing multiple subtasks. Formally, given a query q

as a sample of the user's underlying task $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$ comprising n unknown subtasks, our goal is to produce an ordered set of k key phrases $\mathcal{S} = \{s_1, s_2, \cdots, s_k\}$ with maximum coverage of the subtasks in $\mathcal{T}$ and with minimum redundancy [11].

Most previous works on query recommendation deal with entire queries, which can be problematic, particularly when dealing with rare or unseen queries. Instead, we propose to represent queries at the segment level, by focusing on tasks centered around queries with a named entity, which amount to over 70% of the web search traffic [8]. Inspired by Guo et al. [8], we segment each query into two parts: a named entity e and its associated context c.[1] As a result, we can represent a query log as a weighted bipartite graph $\mathcal{G} = (\mathcal{E}, \mathcal{C}, \mathcal{L})$ where $\mathcal{E}$ denotes the set of entities present in the log, $\mathcal{C}$ the set of contexts, and $\mathcal{L} = \{(e, c) \mid e \in \mathcal{E}, c \in \mathcal{C}\}$ denotes the set of edges corresponding to queries (e, c), weighted by:

$$\ell(e, c) = \frac{\Pr(e, c)}{\Pr(e)\Pr(c)}, \tag{1}$$

where $\Pr(e, c)$ is the probability that both e and c appear in the same query in the query log, $\Pr(e)$ the prior of entity e in the log, and $\Pr(c)$ the prior of context c. This formulation is used so entities or contexts that are popular in the log, but not locally relevant, will receive a lower weight. The resulting graph $\mathcal{G}$ is illustrated in Figure 2(a),[2] with the input query highlighted as a black edge linking the tuple (e_0, c_0). The query recommendation problem then becomes a problem of recommending edges on $\mathcal{G}$, even if the recommended edge is not present in the original log.

4 TRAVERSAL STRATEGIES

We propose three alternative strategies for traversing the graph $\mathcal{G}$ in order to recommend useful and diverse recommendations for task understanding. These strategies are illustrated in Figures 2(b)-(d).

Direct Expansion (DE). Given an input query $q = (e_0, c_0)$ with entity e_0 and context c_0, a simple strategy to recommend queries is to look for other contexts related to e_0. This strategy, called *direct expansion*, is illustrated in Figure 2(b). In the example, given the query "tickets to london," this strategy could return useful recommendations such as "london weather" and "hotels in london," but perhaps not so useful ones such as "history of london." Because we weight edges proportionally to their frequency in the log (see Equation (1)), informational queries such as "history of london" will be demoted in favor of navigational and transactional ones, which are more likely to convey subtasks [13]. Intuitively, this strategy works best in situations where the user is looking for other actions to perform related to the input entity itself.

Syntagmatic Expansion (SE). Because direct expansion promotes alternative contexts related to the input entity e_0, it can lack in diversity. For instance, a user planning a trip may be interested in points-of-interest other than the one represented by e_0 itself. To promote recommendations associated with entities related to e_0, we propose a second strategy, called *syntagmatic expansion.* As illustrated in Figure 2(c), this strategy promotes entities e_i (e.g., "big ben") that are topically related to the input entity e_0 (e.g., "london"). To this end, instead of exploiting topical relationships explicitly stated in the query log (e.g., entities that share many contexts), to attenuate the sparsity problem, we resort to matching entities in a semantic space. In particular, we leverage entity embeddings pretrained on Wikipedia [10][3] and identify the 50 nearest neighbor entities to e_0. Candidate recommendations are then produced by pairing each neighbor entity e_i with its 50 most salient contexts $c_i \in \mathcal{C}$ according to Equation (1). In our example in Figure 2(c), we could pair "big ben" with salient contexts such as "tickets to #" and "# opening hours." Finally, to rank the set of up to $50 \times 50 = 2{,}500$ recommendations, we score each entity-context pair (e_i, c_i) as:

$$f(e_i, c_i) = (\vec{e}_0 \bullet \vec{e}_i) + \ell(e_i, c_i), \tag{2}$$

where $(\vec{e}_0 \bullet \vec{e}_i)$ denotes the dot product between dense vector representations of e_0 and e_i, and $\ell(e_i, c_i)$ is given by Equation (1).

Analogical Expansion (AE). To promote non-trivial recommendations, we propose a third traversal strategy, denoted *analogical expansion.* The intuition behind this strategy is that related entities to e_0 can be identified by looking at analogous relationships involving entities of the same type as e_0. This strategy comprises five steps. First, to identify a set of entities of a similar type as e_0, we select the top 50 entities e_s that share the same context c_0 with the input entity e_0 in $\mathcal{G}$, ranked by $\ell(e_s, c_0)$. In Figure 2(d), "rome" is identified as an entity of the same type as the input entity "london."

As a second step, after retrieving a set of paradigmatically similar entities e_s to e_0, we search $\mathcal{G}$ for entities e_n that frequently follow e_s across sessions. The intuition behind this step is that we can further project the movement $e_s \to e_n$ back to e_0. For instance, if someone searches for a museum after searching for a city similar to the one the user is looking for, we want to move in the same general direction in the embedding space of entities, so we can find another museum (or, more generally, any other entity) related to e_0. In order to decide which movements $e_s \to e_n$ are worth projecting back to e_0 we score each movement according to:

$$f(e_s, e_n) = \ell(e_s, c_0) + \ell(e_s, e_n), \tag{3}$$

where both $\ell(e_s, c_0)$ and $\ell(e_s, e_n)$ are given by Equation (1), except that the latter is overloaded to weight the co-occurrence of two entities within sessions rather than the co-occurrence of an entity and a context within queries. In total, we keep the top 5 entities e_n that follow each entity e_s. In our example in Figure 2(d), "rome" $\to$ "vatican" represents one such frequent movement.

As a third step, to project the identified movements back to the input entity e_0, we perform an analogy operation [10]. Given embeddings for the original entity (e_0), typically similar entity (e_s), and next entity (e_n), we identify a related entity e_i according to $e_i = \vec{e}_0 - \vec{e}_s + \vec{e}_n$, where $\vec{e}_\bullet$ is the dense vector representation of $e_\bullet$. Following our example, "downing street" is selected as a related entity to "london" analogously to how "vatican" is related to "rome."

As a fourth step, the entity e_i selected via analogies must be paired with a suitable context to produce a well-formed recommendation. Given the exploratory nature of analogical expansions, simply choosing the contexts c_i that most frequently co-occur with e_i may cause an undesired topic drift. An alternative could be to

[1]For a query with multiple named entities recognized, only the most salient one is considered, with the remainder of the query retained as its context.
[2]In order to keep the original position of the entity with respect to its context, we replace the surface form of the entity with an out-of-vocabulary placeholder (#).

[3]https://github.com/idio/wiki2vec

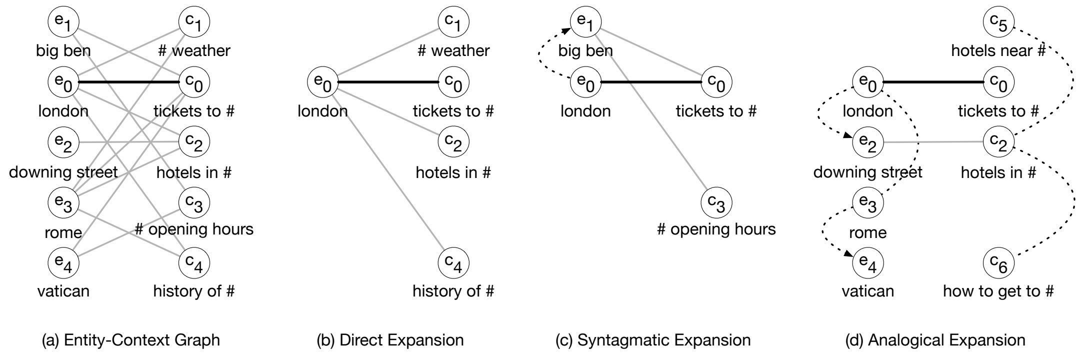

Figure 2: Entity-context graph (a) and different traversal strategies (b-d). A solid black line denotes the input query, whereas solid gray lines denote candidate recommendations. Dashed lines denote inferred associations between entities or contexts.

look for contexts c_i that co-occur with the input entity e_0, which in turn could lead to poor diversity. As a compromise, we train a state-of-the-art neural query recommendation approach [12] on the underlying query log to produce session-sensitive contexts c_i given the input query q, to be matched with the entities e_i.

Because entities and contexts are generated independently in the analogical expansion strategy, we risk recommending semantically incompatible tuples like ("big ben", "mayor of #"). As a fifth and final step, to make sure only meaningful tuples are recommended, we further score each tuple (e_i, c_i) according to:

$$f(e_i, c_i) = \ell(e_s, e_n) + \frac{1}{|C_{e_i}|} \sum_{c_j \in C_{e_i}} (1 - d(\vec{c}_i, \vec{c}_j)), \tag{4}$$

where $\ell(e_s, e_n)$ is the cross-session compatibility (see Equation (3)) of the movement $e_s \rightarrow e_n$ that led to the analogous $e_0 \rightarrow e_i$, $d(\vec{c}_i, \vec{c}_j)$ is the word mover's distance [9] between the embeddings contained in c_i and c_j, and C_{e_i} is the set of all contexts linked to e_i in $\mathcal{G}$. In the next section, we will investigate the effectiveness of these strategies in isolation as well as combined with each other.

5 EXPERIMENTAL EVALUATION

Our experiments aim to assess the effectiveness of our three proposed strategies for query recommendation in task-oriented search.

Test Collection. As test queries, we consider all 50 queries provided by the TREC 2016 Tasks track. Each query includes an explicit reference to the entity e_0 that is the target of the underlying task, as exemplified in Figure 1. The number of subtasks per task ranges from 4 to 14 across the 50 queries. Relevance judgments are available at the subtask-level on a scale from 0 (non-relevant) to 2 (highly relevant). On average, there are 80 unique recommendations judged relevant to at least one subtask per task, as well as 17 recommendations judged highly relevant. As a corpus of candidate recommendations, we use the AOL 2006 query log, semantically annotated using an efficient named entity recognizer for queries [2].

Evaluation Methodology. The ground-truth provided by the TREC 2016 Tasks track comprises a relatively small, non-exhaustive set of judgments, which limits the reusability of this test collection for evaluating new query recommendation strategies [6]. To overcome this limitation, we propose a relaxation scheme to match retrieved recommendations to those in the ground-truth. Under this scheme, a retrieved recommendation s is deemed relevant if its similarity to some ground-truth recommendation g is greater than or equal to a threshold $\theta \in [0, 1]$. As a similarity function, we employ the negative word mover's distance given the bags of word embeddings representing s and g. To assess the utility and diversity of a ranked list of recommendations $\mathcal{S}$, we compute its intent-aware expected reciprocal rank (ERR-IA [7]) at 20 for multiple thresholds $\theta \in [0, 1]$ in steps of 0.1, aggregated into a single point estimate ERR-IA*:

$$\text{ERR-IA}^*@20(\mathcal{S}) = \sum_{\theta} \text{ERR-IA}@20(\mathcal{S}, \theta) \times \exp(\theta), \tag{5}$$

where ERR-IA@20$(\mathcal{S}, \theta)$ leverages the ground-truth relaxed according to the given similarity threshold θ, with the rightmost exponential factor emphasizing the contribution of stricter ground-truths. To ensure our findings are not a mere artifact of this relaxation, we conducted a further, exhaustive evaluation of the best performing strategies with human judges via crowdsourcing. To this end, we used the Figure Eight platform,[4] where three judges assigned scores between 1 and 3 (later averaged) to each recommendation following similar instructions as those used by TREC judges, and with no further information about the origin of each recommendation. We also included known answers as honeypots to filter bad judgments. The augmented ground-truth is available upon request.

Baselines. We consider two baselines representative of classical and neural approaches for query recommendation: Search Shortcuts (SS) [4], implemented using Terrier,[5] and Hierarchical Recurrent Encoder-Decoders (HRED) [12], trained over the AOL query log. Both baselines were deployed with their suggested hyperparameter settings [4, 12]. Neither these baselines nor our proposed strategies require supervision, hence all 50 queries were used for testing.

Experimental Results. We investigate the effectiveness of our proposed strategies, namely, Direct Expansion (DE), Syntagmatic Expansion (SE), and Analogical Expansion (AE), in contrast to two state-of-the-art query recommendation baselines: SS [4] and HRED [12]. To further exploit the potential complementarity of our

[4]https://www.figure-eight.com/
[5]http://terrier.org/

Table 1: ERR-IA*@20 breakdown using relaxed judgments. The number of queries in each bucket is shown in parenthesis. Downward triangles denote significant improvements by the best result (highlighted in bold) in the corresponding column according to a paired t-test with $\alpha = 0.05$.

		All (50)	Head (5)	Torso (12)	Tail (33)
query frequency	HRED	8.26	**10.10**	6.50	8.62
	SS	7.03▼	8.65	4.77▼	7.60▼
	DE	7.06▼	7.62	6.21	7.32▼
	SE	6.41▼	7.49	4.57▼	6.92▼
	AE	5.02▼	4.16	4.09▼	5.74▼
	Mix	**8.71**	7.92	**7.61**	**9.22**
		All (50)	10-16 (16)	17-31 (17)	32-60 (17)
no. of entities	HRED	8.26	7.78	**8.84**	8.13
	SS	7.03 ▼	6.80▼	7.30▼	6.97▼
	DE	7.06▼	6.19▼	7.46 ▼	7.42▼
	SE	6.41▼	6.15▼	6.66▼	6.41▼
	AE	5.02▼	5.02▼	5.03▼	5.02▼
	Mix	**8.71**	**8.23▲**	8.41▼	**9.44▲**
		All (50)	4-6 (21)	7-9 (23)	10+ (6)
no. of subtasks	HRED	8.26	8.98	8.03	6.64
	SS	7.03▼	7.72▼	6.41▼	6.82
	DE	7.06▼	7.67 ▼	6.37▼	7.32
	SE	6.41 ▼	7.21▼	5.64 ▼	6.57
	AE	5.02▼	4.76▼	4.67▼	6.82
	Mix	**8.71**	**9.26**	**8.13**	**8.96**

strategies, we produce a combined strategy (denoted Mix) by aggregating the recommendations produced by all strategies. To penalize duplicate and near-duplicate recommendations contributed by different strategies, the combined strategy employs a ranking model inspired by the idea of maximal marginal relevance (MMR [5]). In particular, each candidate recommendation is rescored based on its similarity to the input query and its dissimilarity to other candidate recommendations. Both similarities compute the word mover's distance between the corresponding word embeddings.

Table 1 shows the results of this investigation in terms of ERR-IA*@20 (see Equation (5)). Considering all 50 queries (first column), we note that, among our individual strategies, DE performs the best, followed by SE and AE. This is somewhat expected, given that DE is the most conservative of the strategies, whereas AE is the most aggressive. Table 1 also shows that, individually, our strategies do not outperform the baselines. However, when combined into the Mix strategy, they outperform both baselines, significantly so in the case of SS. To further assess our strategies, we break down our evaluation for different groups of queries, according to: (i) *query frequency*, aimed to investigate the impact of sparse information, with queries grouped as head (10+ occurrences in the log), torso (1-10), and tail (0); (ii) *number of entities in the ground-truth*, aimed to measure the diversity requirement of different queries, organized in three buckets: queries with 10-16, 17-31, and 32-60 entities; and (iii) *number of subtasks in the ground-truth*, aimed to assess the complexity of the underlying task, with queries also organized in three buckets: those with 4-6, 7-9, and 10+ subtasks.

Table 2: ERR-IA@20 breakdown results across different query buckets using crowdsourced relevance judgments. Symbols are defined analogously to Table 1.

	HRED	SS	Mix
All (50)	1.50	0.96▼	**1.77**
Head (5)	1.40	0.56	**1.54**
Torso (12)	0.98	1.23	**1.43**
Tail (33)	1.70	0.92▼	**1.93**
10-16 entities (16)	1.50	1.09▼	**1.66▲**
17-31 entities (17)	1.30	0.88▼	**1.70▲**
32-60 entities (17)	1.68	0.91▼	**1.95▲**
4-6 subtasks (21)	1.35	0.92	**1.58**
7-9 subtasks (23)	1.61	0.96▼	**1.82**
10+subtasks (6)	1.57	1.09	**2.24**

From Table 1, we further confirm the superiority of DE compared to SE and AE for queries in all groups. Moreover, the combined Mix strategy consistently outperforms SS and HRED in most scenarios. Interestingly, Mix performs the best for hard queries (tail queries, and those that involve a large variety of entities and subtasks). Two exceptions are head queries (5/50 queries with 10+ occurrences in the log) and queries with 17-31 entities in the ground-truth (17/50 queries), where HRED performs the best. To further validate these observations, we submitted the recommendations of Mix, SS, and HRED to additional assessment by human judges via crowdsourcing. The results of this investigation are shown in Table 2. From the table, we first note a generally consistent agreement with the relaxed results reported in Table 1. However, the results attained by Mix are even stronger in this exhaustive judging scenario, with improvements compared to both SS and HRED in all tested query groups. These observations corroborate the effectiveness of our approach for query recommendations for task-oriented search, with improvements over the state-of-the-art for queries with various frequency levels, number of expected relevant entities, and number of underlying subtasks. While the metrics in Tables 1 and 2 are not directly comparable, they show that the results from our relaxed evaluation can be used as a proxy for human judgments.

6 CONCLUSIONS

We proposed three novel strategies for traversing a semantically annotated query log for query recommendation in task-oriented search. Through a comprehensive evaluation extending the experimentation paradigm provided by the TREC Tasks track via relaxed judgments and crowdsourcing, we demonstrated the effectiveness of our strategies and their complementarity in contrast to state-of-the-art query recommendation approaches for queries with different frequency, number of target entities, and of underlying subtasks. In the future, we plan to investigate further traversal strategies for identifying useful recommendations, and to test the suitability of our approach to feed smart assistants.

ACKNOWLEDGMENTS

Work partially funded by project MASWeb (FAPEMIG APQ-01400-14) and by the authors' individual grants from CNPq and FAPEMIG.

REFERENCES

[1] Ricardo A. Baeza-Yates and Alessandro Tiberi. 2007. Extracting semantic relations from query logs. In *KDD*. 76–85.

[2] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking for queries. In *WSDM*. 179–188.

[3] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *WSCD*. 56–63.

[4] Daniele Broccolo, Lorenzo Marcon, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri. 2012. Generating suggestions for queries in the long tail with an inverted index. *Inf. Process. Manage.* 48, 2 (2012), 326–339.

[5] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*. 335–336.

[6] Ben Carterette, Evangelos Kanoulas, Virgil Pavlu, and Hui Fang. 2010. Reusable test collections through experimental design. In *SIGIR*. 547–554.

[7] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *CIKM*. 621–630.

[8] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *SIGIR*. 267–274.

[9] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *ICML*. 957–966.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781 (2013).

[11] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2015. Search result diversification. *Found. Trends. Inf. Retr.* 9, 1 (2015), 1–90.

[12] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*. 553–562.

[13] Manisha Verma, Emine Yilmaz, Rishabh Mehrotra, Evangelos Kanoulas, Ben Carterette, Nick Craswell, and Peter Bailey. 2016. Overview of the TREC Tasks track 2016. In *TREC*.