Mining, Ranking and Recommending Entity Aspects

Ridho Reinanda[†]
r.reinanda@uva.nl

Edgar Meij[‡]
emeij@yahoo-inc.com

Maarten de Rijke[†]
derijke@uva.nl

[†] University of Amsterdam, Amsterdam, The Netherlands
[‡] Yahoo Labs, London, United Kingdom

ABSTRACT

Entity queries constitute a large fraction of web search queries and most of these queries are in the form of an entity mention plus some context terms that represent an intent in the context of that entity. We refer to these entity-oriented search intents as *entity aspects*. Recognizing entity aspects in a query can improve various search applications such as providing direct answers, diversifying search results, and recommending queries. In this paper we focus on the tasks of identifying, ranking, and recommending entity aspects, and propose an approach that mines, clusters, and ranks such aspects from query logs.

We perform large-scale experiments based on users' search sessions from actual query logs to evaluate the aspect ranking and recommendation tasks. In the aspect ranking task, we aim to satisfy most users' entity queries, and evaluate this task in a query-independent fashion. We find that entropy-based methods achieve the best performance compared to maximum likelihood and language modeling approaches. In the aspect recommendation task, we recommend other aspects related to the aspect currently being queried. We propose two approaches based on semantic relatedness and aspect transitions within user sessions and find that a combined approach gives the best performance. As an additional experiment, we utilize entity aspects for actual query recommendation and find that our approach improves the effectiveness of query recommendations built on top of the query-flow graph.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Entity aspects; Query intent; Semantic search

1. INTRODUCTION

With the proliferation of mobile devices, an increasing amount of available structured data, and the development of advanced search result pages, modern-day web search is increasingly geared towards entity-oriented search [2, 26, 29]. A first step and common strategy to address such information needs is to identify entities within queries, commonly known as *entity linking* [24]. Semantic information that is gleaned from the linked entities (such as entity types, attributes, or related entities) is used in various ways by modern search engines, e.g., for presenting an entity card, showing actionable links, and/or recommending related entities [3, 12, 19].

Entities are not typically searched for on their own, however, but often combined with other entities, types, attributes/properties, relationships, or keywords [29]. Such query completions in the context of an entity are commonly referred to as entity-oriented intents or entity aspects [27, 38]. In this paper we study the problem of mining and ranking entity aspects in the context of web search. In particular, we study four related tasks in this paper: (1) identifying entity aspects, (2) estimating the importance of aspects with respect to an entity, (3) ranking entity aspects with respect to a current query and/or user session, and (4) leveraging entity aspects for query recommendation.

The first step in identifying entity aspects involves extracting common queries in the context of an entity and grouping them based on their similarity. We perform this process offline and investigate three matching strategies for clustering queries into entity aspects: *lexical*, *semantic*, and *click-based*. Gathering such entity aspects can already be valuable on its own since they can be used to, e.g., discover bursty or consistent entity intents or to determine entity type-specific aspects [27].

In the next step we rank the obtained entity aspects for each entity in a query-independent fashion using three distinct strategies. This provides us with a mechanism to retrieve the most relevant aspects for a given entity on its own, which, in turn, can be used to, e.g., summarize the most pertinent information needs around an entity or to help the presentation of entity-oriented search results such as customized entity cards on SERPs [2].

The third task that we consider is aspect recommendation. Given an entity and a certain aspect as input, recommend related aspects. This task is motivated by the increasing proliferation of entity-oriented interface elements for web search that can be improved by, e.g., (re)ranking particular items on these elements. Recommending aspects for an entity can also help users discover new and serendipitous information with respect to an entity. We consider two approaches to recommend aspects: *semantic* and *behavioral*. In the semantic approach, relatedness is estimated from a semantic representation of aspects. The behavioral approach is based on the "flow" of aspect transitions in actual user sessions, modeled using an adapted version of the query-flow graph [5, 6, 34].

In our final task we leverage entity aspects for actual query recommendation, i.e., helping users refine their query and/or to help users accomplish a complex search task [11, 22]. Most methods for query recommendation are similar to the behavioral approach men-

SIGIR '15 August 09–13, 2015, Santiago, Chile.

ACM 978-1-4503-3621-5/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2766462.2767724.

tioned above and based on query transitions within sessions. They do not commonly utilize semantic information, however, which may cause distinct but semantically equivalent suggestions. We aim to ameliorate this problem by utilizing the semantic information captured through the entity aspects for query recommendation.

We perform large-scale experiments on both a publicly available and a commercial search engine's query log to evaluate our proposed methods for mining, ranking, and recommending entity aspects, as well as for recommending queries. We perform contrastive experiments using various similarity measures and ranking strategies. We find that entropy-based methods achieve the best performance compared to maximum likelihood and language modeling on the task of entity aspect ranking. Concerning aspect recommendation we find that combining aspect transitions within a session and semantic relatedness give the best performance. Furthermore, we show that the entity aspects can be effectively utilized for query recommendation.

Our main contributions can be summarized as follows:

- We introduce the task of mining, ranking, and recommending entity aspects.
- We provide an in-depth analysis of the mined aspects.
- We propose two approaches to represent aspect relatedness, and utilize them for recommendation.
- We propose a query recommendation method built on top of the query-flow graph.

After a discussion of related work in Section 2, we formalize our task in Section 3. We then detail our approaches to mining, ranking, and recommending entity aspects in Section 4. A detailed account of experiments and data is given in Section 5. We discuss the results of our experiments in Section 6 and conclude in Section 7.

2. RELATED WORK

In this section we review related work around three main topics: query intent mining, leveraging entity aspects for search, and search task identification.

Intent mining deals with identifying clusters of synonymous or strongly related queries based on *intents*, which are typically defined as "the need behind a query." An intent (sometimes also referred to as an *aspect*) is commonly defined as a set of search queries that together represent a distinct information need relevant to the original search query. Methods for identifying intents are typically based on the query itself, results returned by a retrieval algorithm, clicked results, or any other actions by the user. Hu et al. [13] leverage two kinds of user behavior for identifying query "subtopics" (which can be interpreted as intents): one subtopic per search and subtopic clarification by keyword. They propose a clustering algorithm that can effectively leverage the two phenomena to automatically mine the major subtopics of queries. They represent each subtopic as a cluster containing a number of URLs and keywords. Cheung and Li [8] present an unsupervised method for clustering queries with similar intent and producing patterns consisting of a sequence of semantic concepts or lexical items for each intent. They refer to this step of identifying patterns as *intent summarization*. They then use the discovered patterns to automatically annotate queries.

Other related work focuses on extracting attributes from queries, either unsupervised or in the context of entities from a knowledge base. For instance, Li et al. [16] propose a clustering framework with similarity kernels to identify synonymous query intent "templates" for a set of canonical templates. They integrate signals from multiple sources of information and tune the weights in an unsupervised manner. Li et al. [17] on the other hand, solely aim to discover alternative surface forms of attribute values. They propose a compact clustering framework to jointly identify synonyms for a set of attribute values. In a similar vein, Pasca and Van Durme [28] describe a method for extracting relevant attributes or quantifiable properties for various *classes of objects*. They utilize query logs as a source for these. Yin and Shah [38] propose an approach for building a hierarchical taxonomy of generic search intents for a class of named entities. Their proposed approach finds phrases representing generic intents from user queries and organize these phrases into a tree. They propose three methods for tree building: maximum spanning tree, hierarchical agglomerative clustering, and pachinko allocation model. These approaches are based on search logs only. Moving beyond entity types, Lin et al. [19] introduce the notion of active objects in which entity-bearing queries are paired with actions that can be performed on entities. They pose the problem of finding actions that can be performed on entities as the problem of probabilistic inference in a graphical model that captures how an entity-bearing query is generated.

Another body of related work deals with alternative presentations of search results, e.g., based on intents [9]. For instance, Balasubramanian and Cucerzan [2] propose a method to generate entity-specific topic pages as an alternative to regular search results. Similarly, Song et al. [32] present a model to summarize a query's results using distinct aspects. For this they propose "composite queries" that are used for providing additional information for the original query and its aspects. This works by comparatively mining the search results of different component queries. Wu et al. [37] mine latent query aspects based on users' query reformulation behavior and present a system that computes aspects for any new query. Their system combines different sources of information to compute aspects. They first discover candidate aspects for queries by analyzing query logs. They then use a knowledge base to compute aspects for queries that occur less frequently and to group aspects that are semantically related. Finally, Spina et al. [33] explore the task of identifying aspects of an entity given a stream of microblog posts. They compare different IR techniques and opinion target identification methods for automatically identifying aspects.

Search task identification deals with determining the specific task a user is aiming to solve. Such information enables a search engine to, e.g., suggest relevant queries and/or results. Jones and Klinkner [14] formalize the notion of a *search goal* as an atomic information need which results in one or more queries. They propose a method for the automated segmentation of a users' query stream into hierarchical units. While a search goal is atomic, a series of search goals then form a *search missions* (or *complex search tasks*). Lucchese et al. [20] also aim to identify user search tasks within query sessions. They cluster queries in order to find *user tasks*, defined as a set of queries that is aimed towards the same information need. Later they expand user task detection across user sessions [21], similar to so-called task trails and long-term search tasks [18, 36]. Li et al. [15] model the temporal influence of queries within a search session and then use this temporal influence across multiple sessions to identify and label search tasks.

There exists a large body of work on query recommendation, i.e., suggesting follow-up queries to a user, either in an ad hoc fashion or in the context of a user's session or task. Boldi et al. [5] introduces the notion of query-flow graph for query suggestion and Szpektor et al. [34] later expand this model to increase coverage for long tail queries. Bonchi et al. [6] expand it even further to improve coverage. Feild and Allan [10] show that using contextual information can improve query recommendation, as

Table 1: Glossary of the main notation used in this paper.

t	a term
q	a query
e	an entity
s	a query segment, i.e., a sequence of terms
a	an entity aspect, consisting of zero or more query segments
A_e	the set of entity aspects for e
d	time span

long as the previous queries in the context involve a similar or related task. Hassan Awadallah et al. [11] capitalizes on this idea and propose grouping together similar queries and then using them for query recommendation for complex search tasks, similar to task-specific recommendations [22]. Finally, Verma and Yilmaz [35] extract common tasks in the context of an entity to improve retrieval through query expansion and query term prediction. They extract terms frequently appearing with an entity and aggregate this type of information to an entity type level to obtain a dictionary of entity tasks. They evaluate their work through query term prediction and query expansion.

Our work is different in the following major ways. First, we extract entity aspects from query logs specifically. Second, we weight these aspects and assign their importance with respect to an entity on its own in an ad hoc fashion, i.e., without any user, session or query-based information. Third, we learn their relatedness using semantic and behavioral approaches. Finally, we propose an entity aspect-based query recommendation algorithm building upon the query-flow graph.

3. PROBLEM DEFINITION

In this paper, we study three related entity-oriented tasks that are elemental in modern-day entity-oriented web search: *identifying*, *ranking*, and *recommending* entity aspects. Although they build on one another, we propose effective methods for each of them separately since they are essential building blocks for information access applications on their own as well.

In our methods and experiments we employ user interaction log data in the form of queries and clicks. Formally, such logs can be represented as a sequence of events where each event is an action taken by a user. For each event we store a timestamp, a user ID, and the type of action; these are limited to queries and clicks in our current case. Furthermore, the logs are divided into time-ordered sessions, $h \in H$, for each user where we use a common segmentation method and begin a new session after a predefined period of inactivity (30 minutes unless indicated otherwise).

We formulate the first task of *identifying entity aspects* as follows. Given an annotation function $\lambda_e : Q \rightarrow E$ that assigns entities from the set of all entities E to queries, we detect "entity-bearing queries," i.e., queries Q_e containing an entity e. Then, for each entity, we mine a set of entity aspects: $A_e = \{a_1, \ldots, a_m\}$ from Q_e representing the key search tasks in the context of that entity. Table 1 details the main notation we use in this paper. We employ the following definition of search tasks and entity aspects.

> *Given a "search task," defined as an atomic information need resulting in one or more queries, an "entity aspect" is an entity-oriented search task, i.e., a set of queries that represent a common task in the context of an entity, grouped together if they have the same intent.*

Once entity aspects have been identified we turn to ranking them. That is, we estimate the importance of each aspect with respect to the entity in a query-independent fashion and rank them accordingly. The obtained ranking can be interpreted as a distribution of prior probabilities over users' information needs on the entity and can be used on its own to, e.g., prioritize an entity display. For entity aspect recommendation, we recommend related aspects in the context of the entity given an entity-bearing query. We also study this problem in a context-aware setting, incorporating previous queries within the search session. Finally, for query recommendation we drop this restriction and include all queries in a session.

4. METHOD

In this section, we introduce our methods for each of the tasks introduced in the previous section.

4.1 Identifying entity aspects

To mine aspects for an entity, we first need to identify all queries Q_e that contain entity e from the query log using an annotation function λ_e. Since an entity may be referred to in various ways, we need an effective method for identifying entity mentions in web search queries. Since web search queries are typically short and not grammatically correct [31], we rely on a fairly simple method for entity linking that has been shown to obtain strong performance on such texts [23, 24]. In particular, for each query we generate all possible segmentations and link them to Wikipedia articles. Following [24], we use the CMNS method to generate a link for query segment s:

$$CMNS(e, s) = \frac{|H_{s,e}|}{\sum_{e'} |H_{s,e'}|},$$

where $H_{s,e}$ denotes the set of all links with anchor text s which points to target e in Wikipedia. We start with the longest possible query segments and recurse to smaller n-grams in case no entity mention is detected. In case a segment matches multiple entities, we take the most "common" sense, i.e., the one with the highest CMNS score. We do not specifically evaluate the performance of our linking method for queries in this paper. However, in a recent comprehensive comparison on entity linking for queries, CMNS proved to be a very strong unsupervised baseline for this task [4].

Now that we have the set of all queries containing entity e, we remove the mentions of e from the queries and use the remaining segments as *query contexts*. If the query contains more than one entity, we simply consider each entity on its own with the remainder of the query as its context. In this manner we obtain S_e, the set of all context segments which appear with entity e. We then cluster the contexts $s \in S_e$ such that context segments which have the same intent are grouped together. We consider the following features for clustering: *lexical*, *semantic*, and *click* similarity, covering spelling differences/errors, related words/synonyms, and behavioral information. Below we detail the specific methods for each.

Lexical similarity. We compute the lexical similarity between two query contexts using the Jaro-Winkler distance, computed as follows:

$$lex(s_i, s_j) = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_i|} + \frac{m}{|s_j|} + \frac{m-m'}{m}\right) & \text{otherwise,} \end{cases}$$

where m is the number of matching characters, and m' is half the number of transpositions between the two query context segments.

Semantic similarity. To compute the semantic similarity between two query contexts s_i and s_j, we use *word2vec* [25], sum the vectors of each term within the queries, and determine the cosine dis-

tance of the resulting vectors:

$$sem(s_i, s_j) = \frac{z(s_i) \cdot z(s_j)}{|z(s_i)| \cdot |z(s_j)|}, \quad (1)$$

where $z(s)$ is a function that calculates the semantic vector of s.

Click similarity. Beside lexical and semantic similarity, we also utilize click similarity. In particular, for each query context $s \in S_e$ we obtain all clicked hostnames for all queries containing e and s and combine them into a click vector. We then compute the click similarity between two query segments s_i and s_j using their cosine similarity:

$$click(s_i, s_j) = \frac{c_i \cdot c_j}{|c_i| \cdot |c_j|},$$

where c_i and c_j are click vectors of query s_i and s_j, respectively. The final segment similarity score is calculated by taking the maximum value of the similarity scores.

In order to cluster query contexts into entity aspects, we then employ Hierarchical Agglomerative Clustering (HAC) with complete linkage. We flatten the obtained hierarchical clusters so that for every object s_i, s_j belonging to the same cluster, $sim(s_i, s_j) \geq \theta$. By the end of this step, we have obtained the entity aspects A_e in the form of the query context clusters.

4.2 Ranking entity aspects

The main goal of entity aspect ranking is to estimate the importance of each aspect in the context of an entity in a query-independent fashion. Given an entity and its aspects as input, the output of this task is a list of aspects that is ranked according to their pertinence to the entity. We consider three methods for ranking aspects given an entity. The first model is based on maximum likelihood where we reward more frequently occurring aspects:

$$score_{MLE}(a, e) = \frac{\sum_{s \in a} n(s, e)}{\sum_{a'} \sum_{s \in a'} n(s, e)} \quad (2)$$

Here, $n(s, e)$ denotes the number of times query segment s is queried for in the context of e. Note that this method will not simply place clusters with most members at a higher rank, since there might be clusters with few members in which the members occur more frequently than in a large cluster.

The second model uses entropy-based scoring where we reward the most "stable" aspects using different time granularities including months, weeks, and days. For instance, in the case of days we partition the query log into daily chunks and count the number of times completion s is queried for in the context of e on that day. We then determine the entropy:

$$score_{E_d}(a, e) = \sum_{d \in D} P(a|d, e) \log_2 P(a|d, e), \quad (3)$$

where D is the set of all time units and $P(a|d, e)$ the probability of observing any $s \in a$ in the context of e on time interval d (we omit the minus sign in order to make these scores comparable). In another variant we determine the joint entropy, incorporating a factor $p(d|e)$:

$$score_{EJ_d}(a, e) = \sum_{d \in D} P(a, d|e) \log_2 P(a, d|e), \quad (4)$$

where $P(a, d|e)$ is the joint probability of observing any $s \in a$ and time interval d in the context of e.

The third and final model is based on language modeling and aims to rank aspects by how likely they are generated by a statistical language model based on a textual representation of the entity. More formally, we introduce the following three variants:

$$\begin{aligned} score_{LM}(a, e) &= \prod_{s \in a} P(s|\theta_e) \\ score_{LM\text{-}avg}(a, e) &= \frac{1}{|a|} \sum_{s \in a} P(s|\theta_e) \\ score_{LM\text{-}max}(a, e) &= \max_{s \in a}(P(s|\theta_e)), \end{aligned}$$

where $P(s|\theta_e)$ is determined using maximum likelihood estimation with Dirichlet smoothing:

$$P(s|\theta_e) = \prod_{t \in s} P(t|\theta_e) = \prod_{t \in s} \frac{n(t, r(e)) + \mu P(t)}{\sum_{t'} n(t', r(e)) + \mu}.$$

Here, $r(e)$ is the textual representation of e, for which we consider either the entity's Wikipedia article text, LMW, or the frequency-weighted aggregation of all queries leading to a click on the entity's Wikipedia article, LMC. $P(t)$ is the probability of term t estimated from all textual representations of the type at hand. We set μ to the default value of the average document length. Note that the main difference with the MLE method introduced earlier lies in the fact that the LM approaches are based on unigram term probabilities whereas the MLE estimates operate at the segment level.

4.3 Recommending entity aspects

The goal of this task is to recommend aspects related to the entity and aspect currently being queried. Given such an entity and aspect pair as input, the output of this task is a ranked list of aspects. We consider two methods for recommending entity aspects: a *semantic* and a *behavioral* approach. In the semantic approach, we determine the aspects' relatedness based on semantic similarity. In the behavioral approach, we estimate the relatedness from user sessions, inspired by the query-flow graph [5].

We use a graph-based representation for representing entity aspect relatedness. There are two motivations for this decision. First, having a graph-based representation allows the relatedness of the entity aspects to be computed offline. At retrieval time, we retrieve the candidates and rank them based on previously estimated relatedness. Secondly, having this data encoded as graph allows advanced graph-based compression and recommendation techniques to be applied in the future. We construct a different type of graph for each approach: an *aspect-semantic* graph and an *aspect-flow* graph, respectively.

4.3.1 Semantic approach

For the semantic approach we define the aspect-semantic graph for an entity e as an undirected graph $G_{as} = (V, L, w)$ where:

- The nodes are defined as the set of all aspects for e: $V = A_e$,
- $L \in V \times V$ is the set of undirected edges, and
- $w : L \leftarrow (0, 1)$ is a weighting function that assigns a weight w to edges $l_{ij} \in L$.

We construct G_{as} using Algorithm 1 and compute the relatedness between two aspects a_i, a_j using:

$$sem(a_i, a_j) = \frac{z(a_i) \cdot z(a_j)}{|z(a_i)| \cdot |z(a_j)|}, \quad (5)$$

similar to (1). One main difference with (1) is that z is now computed from the mean of the semantic vectors of all query contexts belonging to a. If the relatedness score is above a threshold ϕ, we construct an edge between aspect a_i, and a_j and assign score as weight w_{ij}.

Algorithm 1 Constructing an aspect-semantic graph for e.

Input: Aspect list: A_e
Output: Aspect-semantic graph: G

$G \leftarrow initializeGraph(A_e)$
for each $a_i \in A$ **do**
  for each $a_j \in A$ **do**
    $w \leftarrow sem(a_i, a_j)$
    if $w > \phi$ **then**
      $e \leftarrow createEdge(a_i, a_j, w)$
      $G \leftarrow G \cup e$
    end if
  end for
end for

Algorithm 2 Constructing an aspect-flow graph for e.

Input: Aspect list: A_e, User sessions: H
Output: Aspect-flow graph: G

$G \leftarrow initializeGraph(A_e)$
for each $a_i \in A$ **do**
  for each $a_j \in A$ **do**
    $w \leftarrow adj(H, a_i, a_j)$
    if $w > \varphi$ **then**
      $e \leftarrow createDirectedEdge(a_i, a_j, w)$
      $G \leftarrow G \cup e$
    end if
  end for
end for

4.3.2 Behavioral approach

The second approach is based on the query-flow graph. Formally, we define an aspect-flow graph as a directed graph $G_{af} = (V, L, w)$ where:

- The set of nodes is $V = A_e \cup \{s, t\}$, i.e., the set of all aspects for e plus additional nodes s and t representing a starting state and a terminal state,
- $L \in V \times V$ is the set of directed edges, and
- $w : L \leftarrow (0, 1)$ is a weighting function that assigns a weight w to edges $l_{ij} \in L$.

Here, we estimate the relatedness between query aspects from user sessions. We determine the relatedness from the adjacency of the aspects:

$$adj(H, a_i, a_j) = \sum_{h \in H} countAdjacent(h, a_i, a_j),$$

where $countAdjacent(h, a_i, a_j)$ denotes the frequency of query transitions of any query segment $s \in a_i$ to any segment in a_j found in the user session h, i.e., how often a_j follows a_i. We construct the aspect-flow graph for each entity with Algorithm 2. First, we construct a node for every aspect a that occurs in the user sessions, H. Then, for every pair of aspects (a_i, a_j), we compute their adjacency in H. We create an edge between two aspects a_i and a_j if the adjacency is above a threshold φ. We assign the adjacency count as the weight w_{ij}.

4.3.3 Generating aspect recommendations

We utilize the aspect-semantic graph G_{as} and aspect-flow graph G_{af} to generate recommendations for an input aspect a in the context of entity e. In the first variant, we generate aspect recommendations without a user session's context as detailed in Algorithm 3.

Algorithm 3 Aspect recommendation.

Input: Aspect graph: G, Input aspect: a;
Output: Ranked aspects: R;

$C \leftarrow getCandidatesFromNeighbors(G, a)$
for each $ca \in C$ **do**
  $score[ca] \leftarrow getWeight(G, a, ca)$
end for
$R \leftarrow rankCandidates(score)$

Algorithm 4 Context-aware aspect recommendation.

Input: Aspect graph: G, Input aspect: a, Search context: S
Output: Ranked aspects: R

$C \leftarrow getCandidatesFromNeighbors(G, a)$
for each $ca \in C$ **do**
  $score[ca] \leftarrow getWeight(G, a, ca)$
end for
for each $p \in S$ **do**
  $score_p \leftarrow decay(a, p) * getWeight(G, p, ca)$
  $score[ca] \leftarrow score[ca] + score_p$
end for
$R \leftarrow rankCandidates(score)$

Here, we retrieve candidate recommendations from all nodes adjacent to a in G. For G_{af}, we only retrieve neighboring nodes connected by the outgoing links from a.

We combine the output of both methods to improve the coverage and effectiveness of our recommendations. First, we combine the outputs with a simple round robin strategy, alternating the retrieval of recommendations from the behavioral and the semantic approach, respectively. The intuition is that the semantic method will be able to complement the behavioral method, since it will have higher coverage if constructed with a relatively low threshold ϕ. We also experiment with another combination method: *convex combination*. We retrieve the scores generated by the behavioral and semantic approaches and combine them with a weight λ:

$$score(a, a') = \lambda \cdot flow(a, a') + (1 - \lambda) \cdot semantic(a, a')$$

Since the scores are on a different scale, we perform *min-max normalization* to each score before combining them. Due to our graph construction process there might be cases where either method can not provide any score; for these we simply assign a zero score.

In a variant of this method, we incorporate context-awareness by looking at the previous queries in a user's search session as detailed in Algorithm 4. First, we retrieve the recommendation candidates from the neighbors of a in G. Then we compute initial recommendation scores for each of them and, lastly, we incorporate scores from any previous aspect a' within the search context S, dampened based on their distance:

$$decay(a, a') = \delta^{|a - a'|},$$

where δ is a decay constant, and $|a - a'|$ indicates the distance between a and a'. The distance is the number of aspects queried by the user between a and a' in the current session S.

4.3.4 Generating query recommendations

So far, we have focused on problems and approaches for ranking and recommending aspects involving the same entity. In this section, we detail how we leverage entity aspects for query recommendation in general. That is, recommending other entities, other entity aspects, or regular/non-entity queries for a given query. We

Algorithm 5 Generating query recommendation (QFG+A).

Input: Input query: q, Query graph: G
Output: Ranked recommendations: R

```
for each q ∈ Q do
    q* ← annotateQuery(q);
    n_q ← matchToNode(G, q*);
    S ← getCandidates(G, n);
    for each n_ca ∈ S do
        score[n_ca] ← getWeight(G, n_q, n_ca);
    end for
    R ← rankCandidates(score)
end for
```

complement a state-of-the-art query recommendation method—the query-flow graph [5]—with information from the entity aspects.

We first apply the information from entity aspects when constructing the query-flow graph. We preserve all other queries that are not entity queries, thus forming the query nodes as in a regular query-flow graph. For an entity-bearing query q_e, we link all mentions of an entity e. Next, if the query contains additional query context, we extract the context segment s from q_e. Then, we match s to an appropriate aspect a in the aspect model A_e of e. We perform this matching by finding a which contains s as its cluster member. We collapse different mentions of the same entity into one entity node and collapse semantically-equivalent queries into one entity aspect node. This way, we obtain a "semanticized" query-flow graph.

Lastly, we introduce our recommendation method, detailed in Algorithm 5. For every input query, we perform entity linking of the query to detect entity bearing queries (the *annotateQuery* function). Next, we match an entity query to a node (the *matchToNode* function) with the similar procedure applied during graph construction. Regular queries will be matched straightforwardly. Lastly, we retrieve recommendation candidates from adjacent nodes, scored by their weights in the graph.

5. EXPERIMENTAL SETUP

In this section we detail our experimental setup, including the data we use, the relevance assessments,[1] and the metrics we employ. Our experiments below address the following research questions:

RQ1 When mining entity aspects, how do different similarity measures compare on the task of clustering queries in the context of an entity?

RQ2 How do different aspect ranking methods compare on the task of ranking entity aspects in a query-independent scenario?

RQ3 How do the semantic and behavioral approaches compare on the task of aspect recommendation?

RQ4 Does incorporating context improve aspect recommendation?

RQ5 Can we leverage the semantic information captured through entity aspects to improve the effectiveness of query recommendation built on top of the query-flow graph?

5.1 Experiments

To answer our research questions we set up 4 experiments, which we describe below. In our experiments we test for statistical significance using a paired t-test, indicating significantly better or worse results at the $p < 0.01$ level with $^{\blacktriangle}$ and $^{\blacktriangledown}$ respectively.

Evaluating mining entity aspects. In this experiment, aimed at answering RQ1, we evaluate the quality of the extracted entity aspects by manually evaluating the generated clusters. We use a set of 50 entities sampled from user logs in a stratified fashion. That is, we bias the sample such that more popular entities are more likely to be included. We then extract the query completions for each entity over a period of time from the *dev-contexts* collection (introduced in the next section). To obtain ground truth data we manually cluster the query segments by grouping those that represent the same aspect together. To evaluate the quality of each entity aspect we employ commonly used cluster quality metrics: B-cubed recall (B-recall), precision (B-precision), and F1 (B-F1) [1].

Evaluating ranking entity aspects. The second experiment is aimed at answering RQ2. Since manually evaluating aspect rankings for entities without any explicit query is not straightforward, we resort to automatic evaluation. We propose an automatic evaluation based on what we call "underspecified" entity queries, that is, queries that contain only an entity. We rely on the assumption that a good aspect ranking is one that, on average, best satisfies users that issue such underspecified entity queries. Specifically, we consider sessions that contain an underspecified entity query and aim to predict any subsequent queries that again contain the entity, plus additional query terms.

For this experiment we consider one month of query logs (the *test-aspect-ranking* collection) that is disjoint from any log data used for training). Because of this disjointness there might be aspects that our method is unable to predict, simply because they have not been seen before. This includes spelling variants, reformulations, and new aspects. In our experiments below we do keep them as relevant samples in the evaluation data in order to mimic a real-life setting as closely as possible.

We consider the following setup: we aim to predict the next query a user issues in a session, only considering pairs of adjacent entity-bearing queries in the session where the second query contains the same entity plus additional query terms. We then observe at which position our method ranks this subsequent query and score it accordingly. Since we only have pairs of queries and thus only one relevant suggestion for each, we report on mean reciprocal rank (MRR) and success rate (SR).

Evaluating recommending entity aspects. The third experiment is aimed at answering RQ3 and RQ4. Here, we evaluate the effectiveness of recommending entities and aspects in the context of a user session and constituent queries. We follow a similar evaluation approach to the ranking task above, i.e., we consider the next query in the session as the target to predict. As such we again report on mean reciprocal rank (MRR) and success rate (SR).

Since detecting entity-dominated sessions is not trivial, we simulate them through the following procedure. First, we extend the session demarcation boundary, effectively merging the sessions belonging to the same user within a 3-day timeframe (the *test-aspect-recommendation* collection). Then, we consider the first entity within these extended user sessions as the reference entity and evaluate the recommendation methods by their effectiveness in predicting subsequent aspects of the entity throughout the remainder of the session. This setup reflects recommending related entity aspects for complex search tasks in the context of an entity.

Evaluating query recommendation. Our fourth and final experiment addresses query recommendation and is aimed at answering RQ5. Here we evaluate actual query pair predictions, following the automatic evaluation method from [34]. We sample 1,000,000

[1] Our relevance assessments and editorial guidelines are available at http://ridhorei.github.io/entity-aspects/.

query sessions from the query logs of a commercial search engine (the *test-query-recommendation* collection) and extract pairs of adjacent queries from the sessions. We evaluate this approach in two configurations: looking at all queries within the sessions (*all-pairs*), and using only the first and last queries (*first-last pairs*). Furthermore, we also differentiate between using all query pair occurrences (allowing possible duplicates of popular queries pairs) and using distinct occurrences only. Our main evaluation metrics for this experiment are again mean reciprocal rank (MRR) and success rate (SR). To gain additional insights, we also look at the fraction of correct predictions at different recommendations cut-off levels: 100 and 10.

5.2 Experimental data and settings

We use two sources of data for training and testing, including user logs of the Yahoo web search engine as well as the AOL query logs. From the former we sample a number of datasets. All the development and test datasets that we use are disjoint, i.e., they are sampled from non-overlapping time periods. The *dev-contexts* dataset is a large, 1-year query log sample containing queries that we use to build the full aspect model for our set of entities. The *dev-clicks* dataset is a 1-month sample used to compute click similarity for the context terms. We build our query-flow graph and the aspect-flow graph on the *dev-flow* dataset (a 1-month sample). The test datasets, *test-aspect-ranking*, *test-aspect-recommendation*, *test-query-recommendation*, are all 1-month samples and unseen query logs that are used in our automatic evaluation methods for our second, third, and fourth experiment, respectively. In addition, we also utilize the publicly available AOL dataset in our second experiment. This last dataset includes queries sampled from March 2006 until May 2006.

We define navigational queries as queries that are in the top-40% in terms of the number of pageviews and that also lead to a click on the top search result in at least 40% of the cases. We detect and subsequently discard navigational queries based on this heuristic.

We perform entity linking and context term extraction using the method described in Section 4.1. Below we focus on Wikipedia entities and we leave using other entity repositories for future work. In order to reduce data sparseness we remove entities that occur in less than 100 queries. This results in a set of about 75k entities of interest.

Our approach involves several parameters. The first parameter, θ is the similarity threshold used for clustering. From a preliminary experiment on held-out data, we obtain the optimal value of $\theta = 0.75$. For the minimum relatedness score in the construction of the aspect-semantic graph, we set $\phi = 0.1$. Following the common practice in constructing a query-flow graph [5, 7], we retain only transitions that appear at least two times, thus $\varphi = 1$. After a preliminary experiment, we set $\lambda = 0.85$ when combining the semantic and flow scores. The decay parameter is set to $\delta = 0.85$.

6. EXPERIMENTAL RESULTS

In this section we answer the research questions presented in the previous section.

6.1 Mining aspects

Our first experiment concerns mining entity aspects. We start by evaluating the quality of each cluster and then zoom in on the aspects generated during the mining process. Table 2 presents the results of using different matching strategies to cluster query context terms into entity aspects. Recall that we perform complete-linkage clustering with a parameter θ as threshold for grouping objects.

Table 2: Entity aspect mining results. Significance is tested against the lexical method (row 1).

Method	B-Recall	B-Precision	B-F1
Lexical	0.9164	0.8258	0.8338
Semantic	0.9452$^{\blacktriangle}$	0.7744$^{\blacktriangledown}$	0.8117$^{\blacktriangledown}$
Click	0.8977	0.6666$^{\blacktriangledown}$	0.6880$^{\blacktriangledown}$
Lexical + semantic	0.9216	0.8629$^{\blacktriangle}$	0.8607$^{\blacktriangle}$
Lexical + click	0.8480$^{\blacktriangledown}$	0.8155$^{\blacktriangledown}$	0.7842$^{\blacktriangledown}$
Semantic + click	0.8686$^{\blacktriangledown}$	0.7788$^{\blacktriangledown}$	0.7680$^{\blacktriangledown}$
Lexical + semantic + click	0.8558$^{\blacktriangledown}$	0.8465$^{\blacktriangle}$	0.8098$^{\blacktriangledown}$

Table 3: Entity aspect mining: clustering output for entity *Paris Saint-Germain F.C.*.

Cluster	Context terms
1	*real, real madrid vs, vs real madrid, real madrid*
2	*results*
3	*live, live streaming, live stream*
4	*guingamp*
5	*match*
6	*highlights*
7	*transfert*
8	*2013*
9	*monaco, monaco direct, monaco streaming*
10	*om, regarder om*
11	*streaming, en streaming*
12	*barca, barca vs*
13	*barcelona, barcelone*
14	*barcelona vs*
15	*anderlecht*

First, we look at the results of individual similarity measures. As we can see, using just lexical matching already results in a fairly good B-cubed recall and precision score. This can be explained by the fact that the lexical matching strategy allows minor changes caused by spelling variants or spelling errors, and is successful in performing clustering on these cases. Semantic similarity achieves higher recall at the cost of precision. This means that the clustering method with the current threshold clusters the object aggressively, for example, grouping aspects such as "daughter" and "mother" together. Click similarity has the lowest precision compared to the other two measures for reasons that we will explain below.

Although lexical similarity provides a good start, this strategy fails to group queries that are semantically related. Thus, combining it with semantic similarity improves recall and precision. This combination proves to be the best performing one, compared to using individual measure and all measures.

Adding click similarity with our current strategy does not work well. Upon closer inspection, we find that a lot of unrelated query contexts point to the same host name. For example, contexts related to an entertainer's news are often directed to the same entertainment site. Therefore, combining clicks with other measures tends to brings down the performance, in particular precision.

Table 3 shows sample output generated by our aspect mining method for the entity *Paris Saint-Germain F.C.* (a Parisian soccer club). Our manually created clusters are shown in Table 4. The aspect mining produces more clusters than the ground truth. The method fails to group "barca, barca vs, barcelona, barcelone, barcelona vs." instead making three clusters from them. With such short strings, the pairwise Jaro-Winkler distance between two objects is bigger than the threshold, thus preventing the objects from

Table 4: Entity aspect mining: clustering ground truth for entity *Paris Saint-Germain F.C.*.

Cluster	Context terms
1	*read, real madrid vs, vs real madrid, real madrid*
2	*results*
3	*live, live streaming, live stream, streaming, en streaming*
4	*guingamp*
5	*match*
6	*highlights*
7	*transfert*
8	*2013*
9	*monaco, monaco direct, monaco streaming*
10	*om, regarder om*
11	*barca, barca vs, barcelona, barcelone, barcelona vs*
12	*anderlecht*

being clustered with complete linkage. Also, in some cases the lexical clustering method groups queries that should not be clustered together because they represent different intent/vertical. For instance, two separate clusters ("monaco") and ("monaco direct, monaco streaming") should be created instead of putting them together in a single cluster.

Next, we look at the different types of aspect that occur in the context of our example entity *Paris Saint-Germain F.C.*. Many of the aspects refer to a fairly common *transactional* intent for a football club such as "live streaming". Other sets of intents are *relational*, which concerns the relationship of the topic entity with other entities (in this case, other football clubs). Another set of intents are *categorical*, that is, they deal with type-related intents, e.g., "results," "match" "highlights" and "transfers." The last aspect we observe concerns attempts to find something related to a certain time point, such as "2013".

To conclude this section, we formulate our answer to our first research question RQ1. Combining lexical and semantic similarity measures performs best on the task of clustering query context terms for entity aspect mining and we select this method for the remaining experiments. Integrating click similarity tend to hurt performance, particularly precision.

6.2 Ranking entity aspects

Our second experiment evaluates the importance of each aspect with respect to the entity in a query-independent setting. The results of this experiment is displayed in Tables 5 and 6. From these tables we observe consistent results across the two datasets. First, the simple maximum likelihood approach already performs quite well, thus providing a good baseline. The entropy-based methods, in particular using month as time units, achieve the best performance overall, outperforming maximum likelihood and language modeling approaches. We further observe that the absolute scores on the AOL dataset are lower overall which is mainly due to the disjointness nature of the query logs that we use to mine the aspects with the queries in the AOL logs.

We use different granularities of time-slices, and experiment with two variant of entropy-based methods. For the baseline MLE approach, we simply use the aspect popularity over the whole range of our main query logs that is used for mining (1 year of data). The different granularities do not really show much difference in terms of performance, although computing entropy on the monthly data provides a slight edge.

Table 5: Entity aspect ranking: results on *test-aspect-ranking*. Significance is tested against the MLE baseline (row 1).

	MRR	SR
MLE	0.1931	0.1110
E_{days}	$0.2013^{\blacktriangle}$	$0.1149^{\blacktriangle}$
E_{weeks}	$0.2015^{\blacktriangle}$	$0.1139^{\blacktriangle}$
E_{months}	$0.2031^{\blacktriangle}$	$0.1162^{\blacktriangle}$
EJ_{days}	$0.2020^{\blacktriangle}$	$0.1208^{\blacktriangle}$
EJ_{weeks}	$0.2048^{\blacktriangle}$	$0.1259^{\blacktriangle}$
EJ_{months}	$0.2052^{\blacktriangle}$	$0.1262^{\blacktriangle}$
LMW	$0.0431^{\blacktriangledown}$	$0.0135^{\blacktriangledown}$
LMW_{avg}	$0.0657^{\blacktriangledown}$	$0.0200^{\blacktriangledown}$
LMW_{max}	$0.0583^{\blacktriangledown}$	$0.0170^{\blacktriangledown}$
LMC	$0.0488^{\blacktriangledown}$	$0.0176^{\blacktriangledown}$
LMC_{avg}	$0.0859^{\blacktriangledown}$	$0.0313^{\blacktriangledown}$
LMC_{max}	$0.0755^{\blacktriangledown}$	$0.0259^{\blacktriangledown}$

There are several possible reasons why language modeling does not work well for this task. First, it is the only approach that does not include any query popularity or frequency information. Secondly, what users search for does not always align with what Wikipedia editors may put in a Wikipedia article—which is in line with previous research [37]. This may result in a so-called knowledge base gap where a user is searching for an important fact that is not included on a Wikipedia article yet. In our case, this may result in low scores with the language modeling approach. Lastly, the fact that the language model based approaches are unigram-based, while the other methods are segment-based might also contribute to the lower scores.

We also considered a second experimental variant where we look for entity-bearing query pairs in the whole session. That is, we discard any non-entity bearing queries in between. We find that the scores using this variant are comparable to those reported here.

It is important to note that we compare a different and diverse set of features, with appropriate functions defined for each type of features. However, since the previous step (clustering query contexts into aspects) are kept constant across method, we argue that this comparison is still valuable despite having to compare the combination of features and functions simultaneously. The end-to-end aspect ranking scores should be comparable.

In conclusion, the results show that ranking entity aspects can be done successfully, resulting in sensible absolute MRR and SR scores and we find that entropy-based methods are the best in ranking entity aspects in a query-independent scenario.

6.3 Recommending aspects

Our third experiment evaluates the quality of recommending entity aspects within a session, comparing the semantic and behavioral approaches. The results are shown in Tables 7 and 8. Overall, we see that the behavioral aspect-flow approach outperforms the purely semantic approach. When combined, they give the best performance. In our experiments, we experiment with a round-robin and a convex approach to combine the results.

Upon looking at the graphs created by both methods, we see that the semantic method tends to generate larger graphs, thus attaining more coverage. This is not surprising since the flow graph is only constructed with a single month of data. Despite the sparsity and lack of coverage, the behavioral flow-based approach still manages to outperform the semantic approach, providing better quality rec-

Table 6: Entity aspect ranking: results on the AOL dataset. Significance is tested against the MLE baseline (row 1).

	MRR	SR
MLE	0.0647	0.0340
E_{days}	$0.0710^{\blacktriangle}$	$0.0383^{\blacktriangle}$
E_{weeks}	$0.0712^{\blacktriangle}$	$0.0376^{\blacktriangle}$
E_{months}	$0.0709^{\blacktriangle}$	$0.0376^{\blacktriangle}$
EJ_{days}	$0.0684^{\blacktriangle}$	$0.0383^{\blacktriangle}$
EJ_{weeks}	$0.0692^{\blacktriangle}$	$0.0394^{\blacktriangle}$
EJ_{month}	$0.0692^{\blacktriangle}$	$0.0395^{\blacktriangle}$
LMW	$0.0328^{\blacktriangledown}$	$0.0132^{\blacktriangledown}$
LMW_{avg}	$0.0457^{\blacktriangledown}$	$0.0190^{\blacktriangledown}$
LMW_{max}	$0.0426^{\blacktriangledown}$	$0.0170^{\blacktriangledown}$
LMC	$0.0341^{\blacktriangledown}$	$0.0146^{\blacktriangledown}$
LMC_{avg}	$0.0499^{\blacktriangledown}$	$0.0219^{\blacktriangledown}$
LMC_{max}	$0.0466^{\blacktriangledown}$	$0.0195^{\blacktriangledown}$

Table 7: Aspect recommendation: results. Significance is tested against row 1.

Method	MRR	SR
Aspect-semantic	0.0431	0.0244
Aspect-flow	$0.0602^{\blacktriangle}$	$0.0451^{\blacktriangle}$
Aspect-combined-rr	$0.0674^{\blacktriangle}$	$0.0486^{\blacktriangle}$
Aspect-combined-convex ($\lambda = 0.85$)	$0.0650^{\blacktriangle}$	$0.0465^{\blacktriangle}$

ommendations overall. Both combination approaches succesfully improve over the individual approaches.

As for incorporating user session context (Table 8), we observe that the semantic approach gains a small improvement by incorporating previous queries in the search context. However, the flow-based approach performs slightly worse when previous query is taken into account. This is related to the sparsity/lack of transition data on the flow-based approach. The size of the context have little effect in the current adjancency-based recommendation setup. In conclusion, we find that the behavioral approach is better than the semantic approach for recommending entity aspects, and they can be combined to generate better recommendations.

6.4 Recommending queries

In this section we investigate whether our aspect model can help to complement the query-flow graph for generic query recommendation. Table 9 shows the result of predicting all queries within the sampled user sessions (*all-pairs*). Table 10 shows the results of using the first query as input to predict only the last query of a session (which can be considered as yielding the desired results).

In the *all-pairs* prediction setup, we see that the aspect-based query recommendation method (labeled *QFG+A* in the table) successfully improves upon the baseline query-flow graph in terms of predictions coverage and ranking. The improvement is small, but consistent and significant across the two different configurations of our experiment. Overall, we achieve around 1% improvement on the correct prediction's coverage. For ranking, our method achieves slightly better mean reciprocal rank and average position of the target query (averaged for correct predictions at the top-100). Considering the large number of queries, these improvements are quite substantial. The improvements become more pronounced as we look at the unique query occurrences rather than all occurrences.

We notice consistent results in the *first-last* experiment also. Improvements in terms of prediction coverage are around 1%, while the ranking also shows consistent and significant improvements.

Table 8: Aspect recommendation: results of context aware experiment where m denotes the context size, i.e., the number of queries used as context. Significance is tested against row 1 of each group.

Method	MRR	SR
Aspect-semantic	0.0431	0.0244
CA-aspect-semantic ($m = 3$)	$0.0436^{\blacktriangle}$	$0.0248^{\blacktriangle}$
CA-aspect-semantic ($m = 10$)	$0.0436^{\blacktriangle}$	$0.0248^{\blacktriangle}$
Aspect-flow	0.0602	0.0451
CA-aspect-flow ($m = 3$)	$0.0583^{\blacktriangledown}$	$0.0438^{\blacktriangledown}$
CA-aspect-flow ($m = 10$)	$0.0583^{\blacktriangledown}$	$0.0438^{\blacktriangledown}$

Table 9: Query recommendation: results on the *all-pairs* dataset for each configuration.

	all occurrences	
	QFG	QFG+A
% pairs in all	0.111	0.123
% pairs in top-100	0.076	0.084
% pairs in top-10	0.042	0.047
% pairs in top-1 (SR)	0.015	0.016
MRR	0.024	0.026
avg. position	20.38	20.09
	unique occurrences	
	QFG	QFG+A
% pairs in all	0.108	0.130
% pairs in top-100	0.070	0.088
% pairs in top-10	0.039	0.048
% pairs in top-1 (SR)	0.013	0.015
MRR	0.021	0.026
avg. position	20.57	20.01

7. CONCLUSION

In this paper we have considered common information access tasks in the context of entity-oriented web search. In particular, we have developed and evaluated methods for mining entity aspects, ranking their importance, and recommending them directly or leveraging them for query recommendation. We have done so through linking entities within queries, extracting the query context terms, and clustering them together into entity aspects if they refer to the same intent.

The first step, mining entity aspects involves extracting common queries in the context of an entity and grouping them based on their similarity. We find that combining the *lexical* and *semantic* matching strategies performs best for this task. In the next step we rank the obtained entity aspects for each entity in a query-independent fashion using three strategies: *maximum likelihood*, *entropy*, and *language modeling*. In the maximum likelihood method, we reward more frequently occurring aspects. In the entropy-based methods, we aim to reward aspects that are stable over time. With the language modeling methods, we estimate the probability that the aspect is generated from a statistical unigram language model of the entity. We find that the entropy-based methods yield the best performance. The third task that we consider is aspect recommendation. That is, given an entity and a certain aspect as input, recommend related aspects. For this we consider two approaches, *semantic* and *behavioral*, and find that the latter provides superior results. In our final task we leverage entity aspects for actual query recommendation. Here we normalize a query graph into a semantic query graph, and use the entity aspects as an additional source of infor-

Table 10: Query recommendation: results on the *first-last* dataset for each configuration.

	all occurrences	
	QFG	QFG+A
% pairs in all	0.147	0.165
% pairs in top-100	0.097	0.110
% pairs in top-10	0.055	0.062
% pairs in top-1 (SR)	0.019	0.021
MRR	0.031	0.034
avg. position	19.20	18.98
	unique occurrences	
	QFG	QFG+A
% pairs in all	0.104	0.126
% pairs in top-100	0.062	0.079
% pairs in top-10	0.031	0.041
% pairs in top-1 (SR)	0.009	0.011
MRR	0.016	0.021
avg. position	21.84	20.90

mation for query recommendation. We find that resolving entities and grouping queries into aspects helps to improve query recommendation in a semantic way, by addressing the sparsity of queries and improving the diversity of recommendations.

As to future work, we would like to extend the study in the following directions. First, we would like to experiment with more advanced graph-based compression and recommendation methods for query recommendation. Secondly, we would like to incorporate more features and study the performance of aspect mining in an even larger-scale setting. Next, just as some entity relations are known to be fluent [30], we would like to study the temporality of the entity aspects. Finally, we would like to the see whether the different aspect ranking methods perform differently per entity type, or per different query triggers, distinguishing e.g., actual user inputs with auto-completions and related searches.

Acknowledgments. This research was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, Amsterdam Data Science, the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under project nr 027.012.105, the Yahoo! Faculty Research and Engagement Program, the Microsoft Research PhD program, and the HPC Fund.

8. REFERENCES

[1] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.*, 12(4):461–486, 2009.
[2] N. Balasubramanian and S. Cucerzan. Topic pages: An alternative to the ten blue links. In *IEEE-ICSC 2010*, 2010.
[3] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *ISWC '13*, 2013.
[4] R. Blanco, G. Ottaviano, and E. Meij. Fast and space-efficient entity linking for queries. In *WSDM '15*, 2015.
[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: Model and applications. In *CIKM '08*, 2008.
[6] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. Efficient query recommendations in the long tail via center-piece subgraphs. In *SIGIR '12*, 2012.
[7] I. Bordino, G. De Francisci Morales, I. Weber, and F. Bonchi. From Machu Picchu to rafting the Urubamba River: Anticipating information needs via the entity-query graph. In *WSDM '13*, 2013.
[8] J. C. K. Cheung and X. Li. Sequence clustering and labeling for unsupervised query intent discovery. In *WSDM '12*, 2012.
[9] A. Chuklin, P. Serdyukov, and M. de Rijke. Using intent information to model user behavior in diversified search. In *ECIR '13*, 2013.
[10] H. Feild and J. Allan. Task-aware query recommendation. In *SIGIR '13*, 2013.
[11] A. Hassan Awadallah, R. W. White, P. Pantel, S. T. Dumais, and Y.-M. Wang. Supporting complex search tasks. In *CIKM '14*, 2014.
[12] L. Hollink, P. Mika, and R. Blanco. Web usage mining with semantic analysis. In *WWW '13*, 2013.
[13] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *SIGIR '12*, 2012.
[14] R. Jones and K. L. Klinkner. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *CIKM '08*, 2008.
[15] L. Li, H. Deng, A. Dong, Y. Chang, and H. Zha. Identifying and labeling search tasks via query-based hawkes processes. In *SIGKDD '14*, 2014.
[16] Y. Li, B.-J. P. Hsu, and C. Zhai. Unsupervised identification of synonymous query intent templates for attribute intents. In *CIKM '13*, 2013.
[17] Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Mining entity attribute synonyms via compact clustering. In *CIKM '13*, 2013.
[18] Z. Liao, Y. Song, L.-w. He, and Y. Huang. Evaluating the effectiveness of search task trails. In *WWW '12*, 2012.
[19] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *WWW '12*, 2012.
[20] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *WSDM '11*, 2011.
[21] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.*, 31(3):14:1–14:43, Aug. 2013.
[22] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Modeling and predicting the task-by-task behavior of search engine users. In *OAIR '13*, 2013.
[23] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. de Rijke. Mapping queries to the linking open data cloud: A case study using dbpedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):418–433, 2011.
[24] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM 2012*, 2012.
[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.
[26] P. Pantel and A. Fuxman. Jigs and lures: Associating web queries with structured entities. In *ACL '11*, 2011.
[27] P. Pantel, T. Lin, and M. Gamon. Mining entity types from query logs via user intent modeling. In *ACL'12*, 2012.
[28] M. Pasca and B. Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI'07*, 2007.
[29] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW '10*, pages 771–780, 2010.
[30] R. Reinanda and M. de Rijke. Prior-informed distant supervision for temporal evidence classification. In *Proceedings of COLING 2014*, 2014.
[31] U. Sawant and S. Chakrabarti. Learning joint query interpretation and response ranking. In *WWW '13*, 2013.
[32] W. Song, Q. Yu, Z. Xu, T. Liu, S. Li, and J.-R. Wen. Multi-aspect query summarization by composite query. In *SIGIR '12*, 2012.
[33] D. Spina, E. Meij, M. de Rijke, A. Oghina, M. T. Bui, and M. Breuss. Identifying entity aspects in microblog posts. In *SIGIR '12*, 2012.
[34] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *WWW '11*, 2011.
[35] M. Verma and E. Yilmaz. Entity oriented task extraction from query logs. In *CIKM '14*, 2014.
[36] H. Wang, Y. Song, M.-W. Chang, X. He, R. W. White, and W. Chu. Learning to extract cross-session search tasks. In *WWW '13*, 2013.
[37] F. Wu, J. Madhavan, and A. Halevy. Identifying aspects for web-search queries. *J. Artif. Int. Res.*, 40(1):677–700, 2011.
[38] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *WWW '10*, 2010.