Dynamic Collective Entity Representations for Entity Ranking

David Graus
University of Amsterdam
d.p.graus@uva.nl

Manos Tsagkias
904Labs
manos@904labs.com

Wouter Weerkamp
904Labs
wouter@904labs.com

Edgar Meij
Yahoo Labs
emeij@yahoo-inc.com

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

ABSTRACT

Entity ranking, i.e., successfully positioning a relevant entity at the top of the ranking for a given query, is inherently difficult due to the potential mismatch between the entity's description in a knowledge base, and the way people refer to the entity when searching for it. To counter this issue we propose a method for constructing dynamic collective entity representations. We collect entity descriptions from a variety of sources and combine them into a single entity representation by learning to weight the content from different sources that are associated with an entity for optimal retrieval effectiveness. Our method is able to add new descriptions in real time and learn the best representation as time evolves so as to capture the dynamics of how people search entities. Incorporating dynamic description sources into dynamic collective entity representations improves retrieval effectiveness by 7% over a state-of-the-art learning to rank baseline. Periodic retraining of the ranker enables higher ranking effectiveness for dynamic collective entity representations.

Keywords

Entity ranking; Content representation

1. INTRODUCTION

Many queries issued to general web search engines are related to entities [20]. Entity ranking, where the goal is to position a relevant entity at the top of the ranking for a given query, is therefore becoming an ever more important task [4, 5, 10–12]. In this paper we focus on the scenario in which a searcher enters a query that can be satisfied by returning an entity from a knowledge base (e.g., Wikipedia). Note that this is not the same task as entity linking, where the goal is to identify to which entities a searcher refers in her query.

Entity ranking is inherently difficult due to the potential mismatch between the entity's description in a knowledge base and the way people refer to the same entity when searching for it. When we look at how entities are described, two aspects, context and time, are of particular interest and pose challenges to any solution to entity ranking. Here we explain both aspects.

WSDM 2016, February 22 – 25, 2016, San Francisco, CA, USA

ISBN 978-1-4503-3716-8/16/02...$15.00

DOI: http://dx.doi.org/10.1145/2835776.2835819

Context dependency: Consider the entity `Germany`. A history student could expect this entity to show up when searching for entities related to World War II. In contrast, a sports fan searching for World Cup 2014 soccer results is also expecting to find the same entity. The challenge then becomes how to capture these different contexts for one single entity.

Time dependency: Entities are not static in how they are perceived. Consider `Ferguson, Missouri`, which had a fairly standard city description before the shooting of Michael Brown happened in August 2014. After this event, the entity description of Ferguson changed substantially, reflecting people's interest in the event, its aftermath and impact on the city.

We propose a method that addresses both challenges raised above. First, we use the collective intelligence as offered by a wide range of entity "description sources" (e.g., tweets and tags that mention entities), and we combine these into a "collective entity representation," i.e., a representation that encapsulates different ways of how people refer to or talk about the entity. Consider the example in Figure 1 in which a tweet offers a very different way to refer to the entity `Anthropornis` than the original knowledge base description does. Second, our method takes care of the time dependency by incorporating dynamic entity description sources, which in turn affect the entity descriptions in near real time. Dynamics is part of our method in two ways: (i) we leverage dynamic description sources to expand entity representations, and (ii) we learn how to combine the different entity descriptions for optimal retrieval at specific time intervals. The resulting *dynamic collective entity representations* capture both the different contexts of an entity and its changes over time. We refer to the collection of descriptions from different sources that are associated with an entity as the entity's *representation*. Our method is meant to construct the optimal representation for retrieval, by assigning weights to the descriptions from different sources.

Collecting terms associated with documents and adding them to the document (document expansion) is in itself not a novel idea. Previous work has shown that it improves retrieval effectiveness in a variety of retrieval tasks; see Section 2. However, while information retrieval on the open web is inherently dynamic, document expansion for search has mainly been studied in static, context-independent settings, in which expansion terms from one source are aggregated and added to the collection before indexing [21, 26, 27, 39]. In contrast, our method leverages different dynamic description sources (e.g, queries, social media, web pages), and in this way uses collective intelligence to bridge the gap between the

Figure 1: Entity description of `Anthropornis` in Wikipedia and a tweet with an alternative description of the same entity.

terms used in a knowledge base's entity descriptions and the terms that people use to refer to entities.

To achieve this, we represent entities as fielded documents [24], where each field contains content that comes from a single description source. In a dynamic setting such as our entity ranking setting, where new entity descriptions come in as a stream, learning weights for the various fields in batch is not optimal. Ideally, the ranker continuously updates its ranking model to successfully rank the entities and incorporate newly incoming descriptions. Hence, constructing a dynamic entity representation for optimal retrieval effectiveness boils down to dynamically learning to optimally weight the entity's fields that hold content from the different description sources. To this end we exploit implicit user feedback (i.e., clicks) to retrain our model and continually adjust the weights associated to the entity's fields, much like online learning to rank [16].

Our dynamic collective entity representations generate one additional challenge, which is related to the heterogeneity that exists among entities and among description sources. Popular head entities are likely to receive a larger number of external descriptions than tail entities. At the same time the description sources differ along several dimensions (e.g., volume, quality, novelty). Given this heterogeneity, linearly combining retrieval scores (as is commonly done in structured retrieval models) proves to be suboptimal. We therefore extend our method to include features that enable the ranker to distinguish between different types of entities and stages of entity representations.

We address these three research questions:

RQ1 Does entity ranking effectiveness increase by using dynamic collective entity representations? We compare a baseline entity ranking method based on the information in the knowledge base only to our method that incorporates additional description sources (web anchors, queries, tags, and tweets).

RQ2 Does entity ranking effectiveness increase when employing additional field and entity importance features? We answer this question by incorporating field and entity importance features to our knowledge base-only baseline ranker and our proposed method, and compare their performance.

RQ3 Does entity ranking effectiveness increase when we continuously learn the optimal way to combine the content from different description sources? We compare a static entity ranking baseline that is trained once at the start, to our proposed method that is retrained at regular intervals.

The main contribution of this paper is a novel approach to constructing dynamic collective entity representations, which takes into account the temporal and contextual dependencies of entity descriptions. We show that dynamic collective entity representations better capture how people search for entities than their original knowledge base descriptions. In addition, we show how field importance features better inform the ranker, thus increasing retrieval effectiveness. Furthermore, we show how continuously updating the ranker enables higher ranking effectiveness. Finally, we perform extensive analyses of our results and show that incorporating dynamic signals into the dynamic collective entity representation enables a better matching of users' queries to entities.

2. RELATED WORK

Related work comes in three flavors: entity ranking, document expansion and fielded or structured information retrieval.

2.1 Entity ranking

Recent years have shown an increase in entity ranking research. Main causes for this were the introductions of the INEX Entity Ranking track [10–12] and the TREC Entity track [4, 5]. Exploiting content and structure of knowledge bases is common practice in entity ranking, for example by including anchor texts and inter-entity links [33], category structure [3, 32], entity types [18], or internal link structure [38]. More recently, researchers have also started to focus on using query logs for entity ranking. Billerbeck et al. [8] use query logs of a web search engine to build click and sessions graphs and walk the graph to answer entity queries. Mottin et al. [29] use query logs to train an entity ranking system using different sets of features. Unlike us, the authors assume a static setting (i.e., no online learning) and do not take into account external sources for entity representation. Hong et al. [17] follow a similar line of thought and enrich their knowledge base using linked web pages and queries from a query log. Again, the authors assume a static setting in which web pages and queries are added to the knowledge base before any ranking experiments are performed. One of the few initial attempts to bring in time in entity ranking is a position paper by Balog and Nørvåg [2], who propose temporally-aware entity retrieval, in which temporal information from knowledge bases is required.

2.2 Document expansion

Singhal and Pereira [34] are one of the first to use document expansion in a (speech) retrieval setting, motivated by the vocabulary mismatch introduced by errors made by automatic speech transcription. Ever since, using external sources for expanding document representations has been a popular approach to improve retrieval effectiveness. In particular, it was shown that anchors can improve ad-hoc web search [14, 26, 39, 40].

Kemp and Ramamohanarao [19] state that document transformation using search history, i.e., adding queries that lead to the document to be clicked, brings documents closer to queries and hence improves retrieval effectiveness. Similarly, Xue et al. [41] study the use of click-through data by adding queries to clicked document representations. In this case, the click-through data includes a score that is derived from the number of clicks the query yields for a single document. Gao et al. [15] follow a similar approach, but add smoothing to click-through data to counter sparsity issues. Amitay et al. [1] study the effectiveness of query reformulations for document expansion by appending all queries in a reformulation session to the top-k returned documents for the last query. Scholer et al. [32] propose a method to either add additional terms from associated queries to documents or replace the original content with these associated queries, all with the goal of providing more accurate document representations.

Looking at other sources for expansion, Bao et al. [6] improve

web search using social annotations (tags). They use the annotations both as additional content as well as popularity measure. Lee and Croft [21] explore the use of social anchors (i.e., content of social media posts linking to a document) to improve ad hoc search. Noll and Meinel [30] investigate of a variety of "metadata" sources, including anchors, social annotations, and search queries. They show that social annotations are concise references to entities and outperform anchors in several retrieval tasks. Efron et al. [13] show that document expansion can be beneficial when searching for very short documents (tweets).

2.3 Fielded retrieval

A common approach to incorporate the document expansions into a document is to create fielded documents [31, 42]. Based on fielded documents, a variety of retrieval methods have been proposed. Robertson et al. [31] introduce BM25F, the fielded version of BM25, which linearly combines query term frequencies over different field types. Broder et al. [9] propose an extension to BM25F, taking into account term dependencies. Svore and Burges [35] use a machine learning approach for learning BM25 over multiple fields, the original document fields (e.g., title and body) and so-called "popularity" fields (e.g., anchors, query-clicks). Macdonald et al. [24] compare the linear combination of term frequencies before computing retrieval scores to directly using retrieval scores in the learning to rank setting and show that it is hard to determine a clear winner.

What sets our work apart from the aforementioned work is that we deal with dynamic fields with continually changing content, and study continually adapting entity representations and rankers. Our method is required to reweigh fields continuously to reflect changes in the fields' content. Continually (re-)tuning parameters of fielded retrieval methods such as BM25F when documents in the index change is exceedingly expensive, rendering these methods unsuitable for this scenario.

3. DYNAMIC COLLECTIVE ENTITY REPRESENTATIONS

3.1 Problem statement

The problem of entity ranking is: given a query q and a knowledge base KB populated with entities $e \in E$, find the best matching e that satisfies q. Both e and q are represented in some (individual or joint) feature space that captures a range of dimensions, which characterize them individually (e.g., content, quality) and jointly (e.g., relationships through click logs).

The problem itself is a standard information retrieval problem, where the system needs to bridge the gap between the vocabulary used in queries and documents. This is a long-standing but still open problem that has been tackled from many perspectives. One is to design better similarity functions, another is to develop methods for enhancing the feature spaces. Our method shares characteristics with both perspectives, as we will now explain.

3.2 Approach

Our approach to the entity ranking problem consists of two interleaved steps. First, we use external description sources (described in §3.3) to expand entity representations and reduce the vocabulary gap between queries and entities. We do so by representing entities as fielded documents, where each field corresponds to content that comes from one description source. Second, we train a classification-based entity ranker, that employs different types of features to learn to weight and combine the content from each field of the entity for optimal retrieval (§3.4).

External description sources may continually update and change the content in the entity's fields, through user feedback (i.e., clicks) following an issued query, or when users generate content on external description sources that is linked to a KB entity (e.g., Twitter, Delicious). Consequently, the feature values that represent the entities change, which may invalidate previously learned optimal feature weights and asks for continuously updating the ranking model.

Table 1: Summary of the nine description sources we consider: Knowledge Base entity descriptions (KB), KB anchors, KB redirects, KB category titles, KB inter-hyperlinks, queries, web anchors, tweets, and tags from Delicious.

Data source	Size	Period	Affected entities
Static expansion sources			
KB	4,898,356	August 2014	–
KB anchors	15,485,915	August 2014	4,361,608
KB redirects	6,256,912	August 2014	N/A
KB categories	1,100,723	August 2014	N/A
KB inter-links	28,825,849	August 2014	4,322,703
Dynamic expansion sources			
Queries	47,002	May 2006	18,724
Web anchors	9,818,004	2012	876,063
Twitter	52,631	2011–2014	38,269
Delicious	4,429,692	2003–2011	289,015

3.3 Description sources

To construct dynamic collective entity representations we use two types of description sources: (i) a knowledge base (KB) from which we extract the initial entity representations, and (ii) a set of external description sources that we use to expand the aforementioned entity representation.

We differentiate between external description sources that are non-timestamped (*static*) and ones that are timestamped (*dynamic*). Non-timestamped sources are those where no time information is available, and sources that are not inherently dynamic, e.g., web archives and aggregates over archives like web anchors. Timestamped external description sources are sources whose content is associated with a timestamp, and where the nature of the source is inherently dynamic or time-dependent, e.g., tweets or query logs. We describe each type of external description source below, the number of total descriptions, and affected entities, and we provide a summary in Table 1.

Knowledge base. The knowledge base that we use as our initial index of entities, and which we use to construct the initial entity representations, is a snapshot of Wikipedia from August 3, 2014 with 14,753,852 pages. We filter out non-entity pages ("special" pages such as category, file, and discussion pages), yielding 4,898,356 unique entities.

Static description sources.

Knowledge base. Knowledge base entities have rich metadata that can be leveraged for improving retrieval [3, 32, 33]. We consider four types of metadata to construct the KB entity representations: (i) anchor text of inter-knowledge base hyperlinks, (ii) redirects, (iii) category titles, and (iv) titles of entities that are linked from and to each entity. Editorial conventions and Wikipedia's quality control ensures these expansions to be of high quality.

Web anchors. Moving away from the knowledge base itself, the

web provides rich information on how people refer to entities leading to tangible improvements in retrieval [39]. We extract anchor texts of links to Wikipedia pages from the Google Wikilinks corpus.[1] We collect 9,818,004 anchor texts for 876,063 entities. Web anchors differ from KB anchors as they can be of lower quality (due to absence of editorial conventions) but also of much larger volume. While in theory web anchors could be associated with timestamps, in a typical scenario they are aggregated over large archives, where extracting timestamps for diverse web-pages is non-trivial.

Dynamic description sources.

Twitter. Mishne and Lin [27] show how leveraging terms from tweets that do not exist in the pages linked to from tweets can improve retrieval effectiveness of those pages. We follow a similar approach and mine all English tweets that contain links to Wikipedia pages that represent the entities in our KB. These are extracted from an archive of Twitter's sample stream, spanning four years (2011–2014), resulting in 52,631 tweets for 38,269 entities.

Delicious. Social tags are concise references to entities and have shown to outperform anchors in several retrieval tasks [30]. We extract tags associated with Wikipedia pages from SocialBM0311[2] resulting in 4,429,692 timestamped tags for 289,015 entities.

Queries. We use a publicly available query log from MSN sampled between May 1 and May 31, 2006, consisting of 15M queries and their metadata: timestamps and URLs of clicked documents. We keep only queries that result in clicks on Wikipedia pages that exist in our snapshot of Wikipedia, resulting in 47,002 queries associated with 18,724 uniquely clicked entities. We hold out 30% of the queries for development (e.g., parameter tuning, feature engineering; 14,101 queries) and use 70% (32,901 queries) for testing. Our use of queries is twofold: (i) queries are used as input to evaluate the performance of our system, and also (ii) as external description source, to expand the entity description with the terms from a query that yield a click on an entity. While this dual role of queries may promote head entities that are often searched, we note that "headness" of entities differs across description sources, and even tail entities may benefit from external descriptions (as illustrated by the `Anthropornis` example in Fig. 1).

3.4 Adaptive entity ranking

The second step in our method is to employ a supervised entity ranker that learns to weight the fields that hold content from the different description sources for optimal retrieval effectiveness. Two challenges arise in constructing collective dynamic entity representations.

Heterogeneity. External description sources exhibit different dynamics in terms of volume and quality of content [22], and differences in number and type of entities to which they link (see, e.g., Table 1). This heterogeneity causes issues both within entities, since different description sources contribute different amounts and types of content to the entity's fields, and between entities, since popular entities may receive overall more content from external description sources than tail entities.

Dynamicness. Dynamic external description sources cause the entity's descriptions to change in near real-time. Consequently, a static ranking model cannot capture the evolving and continually changing index that follows from our dynamic scenario, hence we employ an adaptive ranking model that is continuously updated.

[1] https://code.google.com/p/wiki-links/

[2] http://www.zubiaga.org/datasets/socialbm0311/

4. MODEL

In the following section we describe our supervised ranking approach, by first explaining the entity representation, the set of features we employ for learning an optimal representation for retrieval, and finally the supervised method.

4.1 Entity Representation

To deal with dynamic entity representations, which are composed of content from different external description sources, we model entities as fielded documents:

$$e = \{\overline{f}^e_{title}, \overline{f}^e_{text}, \overline{f}^e_{anchors}, \ldots, \overline{f}^e_{query}\}. \quad (1)$$

Where $\overline{f}^e$ corresponds to the field term vector that represents e's content from a single source (denoted in subscript). We refer to this collection of field term vectors as the entity's *representation*.

The fields with content from dynamic description sources may change over time. We refer to the process of adding an external description source's term vector to the entity's corresponding field term vector as an *update*. To model these dynamically changing fields, we discretize time (i.e., $T = \{t_1, t_2, t_3, \ldots, t_n\}$), and define *updating* fields as:

$$\overline{f}^e_{query}(t_i) = \overline{f}^e_{query}(t_{i-1}) + \begin{cases} \overline{q}, & \text{if } e_{clicked} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$\overline{f}^e_{tweets}(t_i) = \overline{f}^e_{tweets}(t_{i-1}) + \overline{tweet}_e \quad (3)$$

$$\overline{f}^e_{tags}(t_i) = \overline{f}^e_{tags}(t_{i-1}) + \overline{tag}_e. \quad (4)$$

In equation 2, $\overline{q}$ represents the term vector of query q that is summed element-wise to e's query field term vector ($\overline{f}^e_{query}$) at time t_i, if e is clicked by a user that issues q. In equation 3, $\overline{tweet}_e$ represents the field term vector of a tweet that contains a link to the Wikipedia page of e, which also gets added element-wise to the corresponding field ($\overline{f}^e_{tweets}$). Finally, in equation 4, $\overline{tag}_e$ is the term vector of a tag that a user assigns to the Wikipedia page of e.

To estimate e's relevance to a query q given the above-described representation one could e.g., linearly combine retrieval scores between q's term vector $\overline{q}$ and each $\overline{f} \in e$. However, due to the heterogeneity that exists both between the different fields that make up the entity representation, and between different entities (described in §3.4), linearly combining similarity scores may be suboptimal [31], and hence we employ a supervised single-field weighting model [24]. Here, each field's contribution towards the final score is individually weighted, through learned field weights from implicit user feedback.

4.2 Features

To learn the optimal entity representation for retrieval, we employ three types of features that express field and entity importance: first, *field similarity* features are computed per field and boil down to query–field similarity scores. Next, *field importance* features, likewise computed per field aim to inform the ranker of the status of the field at that point in time (i.e., to favor fields with more and novel content). Finally, we employ *entity importance* features, which operate on the entity level and aim to favor recently updated entities.

4.2.1 Field similarity

The first set of features model the similarity between a query and a field, which we denote as ϕ_{sim}. For query–field similarity,

we compute TF×IDF cosine similarity. We define

$$\phi_{sim}(q, \overline{f}, t_i) = \sum_{w \in q} n(w, \overline{f}(t_i)) \cdot \log \frac{|C^{t_i}|}{|\{\overline{f}(t_i) \in C^{t_i} : w \in \overline{f}(t_i)\}|}, \tag{5}$$

where w corresponds to a query term, $n(w, \overline{f}(t_i))$ is the frequency of term w in field f at time t_i. C^{t_i} is the collection of fields at time t_i, and $|\cdot|$ indicates set cardinality. More elaborate similarity functions can be used, e.g., BM25(F), however, we choose a parameterless similarity function that requires no tuning. This allows us to directly compare the contribution of the different expansion fields without having additional factors play a role, such as length normalization parameters, which affect different fields in non-trivial ways.

4.2.2 *Field importance*

The next set of features is also computed per field, ϕ_p is meant to capture a field's *importance* at time t_i:

$$\phi_p(\overline{f}(t_i), e) = S(\overline{f}(t_i), e). \tag{6}$$

We instantiate four different field importance features. First, we consider two ways to compute a field's length, either in terms (7) or in characters (8):

$$\phi_{p_1}(\overline{f}(t_i), e) = |\overline{f}(t_i)| \tag{7}$$

$$\phi_{p_2}(\overline{f}(t_i), e) = \sum_{w \in \overline{f}(t_i)} |w| \tag{8}$$

The third field importance scoring function captures a field's novelty at time t_i, to favor fields that have been updated with previously unseen, newly associated terms to the entity (i.e., terms that were not in the original entity representation at t_0):

$$\phi_{p_3}(\overline{f}(t_i), e) = |\{w \in \overline{f}(t_i) : w \notin \overline{f}(t_0)\}|. \tag{9}$$

The fourth field importance scoring function expresses whether a field has undergone an update at time t_i:

$$\phi_{p_4}(\overline{f}, t_i, e) = \sum_{j=0}^{i} += \begin{cases} 1, & \text{if } update(\overline{f}(t_j)) \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where $update(\overline{f}(t_j))$ is a Boolean function indicating whether field f was updated at time j, i.e., we sum from t_0 through t_i and accumulate updates to the fields.

4.2.3 *Entity importance*

The feature ϕ_I models the entity's importance. We compute the time since the entity last received an update to favor recently updated entities:

$$\phi_I(e, t_i) = t_i - \max_{\overline{f} \in e} time_{\overline{f}}, \tag{11}$$

Here, t_i is the current time, and $time_{\overline{f}}$ is the update time of field $\overline{f}$. $\max_{\overline{f}}$ corresponds to the timestamp of the entity's field that was most recently updated.

4.3 Machine learning

Given the features explained in the previous section, we employ a supervised ranker to learn the optimal feature weights for retrieval;

$$\Omega = (\omega_e, \omega_{\overline{f}_{title}}, \omega_{p1_{title}}, \omega_{p2_{title}}, \ldots, \omega_{\overline{f}_{text}}, \omega_{p1_{text}}, \ldots)$$

Here, Ω corresponds to the weight vector, which is composed of individual weights (ω) for each of the field features (similarity and importance) and the entity importance feature.

To find the optimal Ω, we train a classification-based re-ranking model, and learn from user interactions (i.e., clicks). The model employs the features detailed in the previous section, and the classification's confidence score is used as a ranking signal. As input, the ranker receives a feature vector (x), extracted for each entity-query pair. The associated label y is positive (1) for the entities that were clicked by a user who issued query q. We define x as

$$x = \{\phi_{sim_1}, \phi_{p1_1}, \phi_{p2_1}, \phi_{p3_1}, \phi_{p4_1}, \ldots, \\ \phi_{sim_{|e|}}, \phi_{p1_{|e|}}, \phi_{p2_{|e|}}, \phi_{p3_{|e|}}, \phi_{p4_{|e|}}, \phi_I\} \tag{12}$$

Where $|e|$ corresponds to the number of fields that make up the entity representation ($\overline{f} \in e$).

See Algorithm 1 for an overview of our machine learning method in pseudo-code. As input, our supervised classifier is given a set of $\langle q, e, L\rangle$-tuples (see l.4). These tuples consist of a query (q), a candidate entity (e), and a (binary) label (L): positive (1) or negative (0). Given an incoming query, we first perform top-k retrieval (see §5.2 for details) to yield our initial set of candidate entities: $\mathcal{E}_{candidate}$ (l.8). For each entity, we extract the features which are detailed in Section 4.2 (l.12) and have the classification-based ranker R output a confidence score for e belonging to the positive class, which is used to rank the candidate entities (l.13). Labels are acquired through user interactions, i.e., entities that are in the set of candidate entities and clicked after issuing a query are labeled as positive instances (l.16–22), used to retrain R (l.24). Finally, after each query, we allow entities to be updated by dynamic description sources: tweets and tags (l.26), we provide more details in §5.1.

Algorithm 1 Pseudo-algorithm for learning optimal Ω^T.

Require: Ranker R, Knowledge Base KB, Entities $\mathcal{E}$

1: $\mathcal{E} \longleftarrow \{e_1, e_2, \ldots, e_{|KB|}\}$
2: $e \longleftarrow \{\overline{f}_{title}, \overline{f}_{anchors}, \ldots, \overline{f}_{text}\}$
3:
4: $\mathcal{L} = \{\langle q_1, e_1, \{0,1\}\rangle, \langle q_1, e_2, \{0,1\}\rangle, \ldots, \langle q_n, e_m, \{0,1\}\rangle\}$
5: $R \longleftarrow$ Train($\mathcal{L}$)
6:
7: **while** q **do**
8: $\quad\mathcal{E}_{candidate} \longleftarrow$Top-$k$ retrieval(q)
9: $\quad\mathcal{E}_{ranked} \longleftarrow$ []
10:
11: $\quad$**for** $e \in \mathcal{E}_{candidate}$ **do**
12: $\quad\quad\phi_e \longleftarrow$ Extract features(e)
13: $\quad\quad\mathcal{E}_{ranked} \longleftarrow$ Classify(R, ϕ_e)
14: $\quad$**end for**
15:
16: $\quad e_{clicked} \longleftarrow$ Observe click($q, \mathcal{E}$)
17: $\quad$**if** $e_{clicked} \in \mathcal{E}_{candidate}$ **then**
18: $\quad\quad\mathcal{L} \longleftarrow \mathcal{L} \cup \{\langle q, e_{clicked}, 1\rangle\}$
19: $\quad\quad e_{clicked} \longleftarrow e_{clicked} \cup \{text_q\}$
20: $\quad$**else**
21: $\quad\quad\mathcal{L} \longleftarrow \mathcal{L} \cup \{\langle q, e_{clicked}, 0\rangle\}$
22: $\quad$**end if**
23:
24: $\quad R \longleftarrow$ Train($\mathcal{L}$)
25: $\quad$**for** $e \in \mathcal{E}$ **do**
26: $\quad\quad e \longleftarrow e \cup \{\overline{f}_{e,tweet_1}, \overline{f}_{e,tag_1}, \ldots, \overline{f}_{e,tweet_i}, \overline{f}_{e,tag_j}\}$
27: $\quad$**end for**
28: **end while**

5. EXPERIMENTAL SETUP

In this section we start by describing our experiments and how they allow us to answer the research questions raised in Section 1, and then, we present our machine learning setting and describe our evaluation methodology.

Experiment 1. To answer our first research question, *does entity ranking effectiveness increase using dynamic collective entity representations?*, we compare our proposed dynamic collective entity ranking method to a baseline that only incorporates KB fields for entity ranking: KBER (Knowledge Base Entity Representations). We restrict both the baseline and our Dynamic Collective Entity Representation (DCER) method to the set of field similarity features (§4.2.1), which we denote as KBER_{sim} and DCER_{sim}. This allows us to provide clear insights into the contribution of the fields' content in ranking effectiveness. In addition, we perform an ablation study and compare the similarity-baseline to several approaches that incorporate content from a single external description source (denoted KB+source_{sim}).

Experiment 2. We address RQ2, *does entity ranking effectiveness increase when employing field and entity features*, by comparing the KBER_{sim} baseline that only incorporates field similarity features, to the KBER baseline that incorporates the entity and field importance features, and to our DCER method.

Experiment 3. Finally, to answer our third research question, *does entity ranking effectiveness increase when we continuously learn the optimal entity representations?* We compare our proposed entity ranking system DCER to its non-adaptive counterpart (DCER_{na}), that we do not periodically retrain. Contrasting the performance of this non-adaptive system with our adaptive system allows us to tease apart the effects of adding more training data, and the effect of the additional content that comes from dynamic external description sources. Here too, we include an ablation study, and compare to non-adaptive approaches that incorporate content from a single external description source (denoted KB+source_{na}).

Baselines. Due to our focus on dynamic entity representations and adaptive rankers, running our method on datasets from seemingly related evaluation campaigns such as those in TREC and INEX is not feasible. We are constrained by the size of datasets, i.e., we need datasets that are sufficiently large (thousands of queries) and time-stamped, which excludes the aforementioned evaluation campaigns, and hence direct comparison to results obtained there. Furthermore, employing existing fielded retrieval methods such as BM25F as baselines is not feasible either, as they are exceedingly expensive to retrain online, as discussed in Section 2.

For these reasons we consider the following supervised baselines in our experiments; KBER_{sim} is an online learning classification-based entity ranker, that employs field similarity features on entity representations composed of KB description sources (i.e., title, text, categories, anchors, redirects and links fields). KBER is the same baseline system, extended with the full set of features (described in §4.2). Finally, DCER_{na} is a non-adaptive baseline: it incorporates all external description sources, and all features as our proposed DCER method, but does not periodically retrain.

5.1 Data alignment

In our experiments we update the fields that jointly represent an entity with external descriptions that come in a streaming manner, from a range of heterogeneous external description sources. This assumes that all data sources run in parallel in terms of time. In a real-world setting this assumption may not always hold as systems need to integrate historical data sources that span different time periods and are of different size for bootstrapping. We simulate this scenario by choosing data sources that do not originate from the same time period nor span the same amount of time (see also Table 1). To remedy this, we introduce a method for time-aligning all data sources (which range from 2009 to 2014) to the timeline of the query log (which dates from 2006). We apply a *source-time* transformation to mitigate the dependence of content popularity on time and when it was created [36, 37]. Each query is treated as a time unit, and we distribute the expansions from the different sources over the queries, as opposed to obeying the misaligned timestamps from the query log and expansion sources.

To illustrate: given n queries and a total of k items for a given expansion source, after each query we update the entity representations with $\frac{n}{k}$ expansions of that particular expansion source. In our dataset we have 32,901 queries, 52,631 tweets, and 4,429,692 tags. After each query we distribute 1 query, 2 tweets, and 135 tags over the entities in the index. Mapping real time to source time evenly spreads the content of each data source within the timespan of the query log. This smooths out bursts but retains the same distribution of content over entities (in terms of, e.g., entity popularity). The diminishing effect on burstiness is desirable in the case of misaligned corpora, as bursts are informative of news events, which would not co-occur in the different data sources. Although our re-aligned corpora cannot match the quality of real parallel corpora, our method offers a robust lower bound to understand the utility of collective dynamic entity representations of our method. We reiterate that the goal of this paper is to study the effect of dynamic entity representations, and adapting rankers, not to leverage temporal features.

5.2 Machine learning

We apply machine learning for learning how to weight the different fields that make up an entity representation for optimal retrieval effectiveness. In response to a query, we first generate an initial set of candidate entities by retrieving the top-k entities to limit the required computational resources for extracting all features for all documents in the collection [23]. Our top-k retrieval method involves ranking all entities using our similarity function (described in §4.2.1), where we collapse the fielded entity representation into a single document.[3] We choose Random Forests as our machine learning algorithm because it has proven robust in a range of diverse tasks (e.g., [28]), and can produce confidence scores that we employ as ranking signal. In our experiments, we set the number of trees to 500 and the number of features each decision tree considers for the best split to $\sqrt{|\Omega|}$.

5.3 Evaluation

For evaluating our method's adaptivity and performance over time, we create a set of incremental time-based train/test splits as in [7]. We first split the query log into K chunks of size N: $\{C_1, C_2, C_3, \ldots, C_K\}$. We then allocate the first chunk (C_1) for training the classifier and start iteratively evaluating each succeeding query. Once the second chunk of queries (C_2) has been evaluated, we expand the training set with it and retrain the classifier. We then continue evaluating the next chunk of queries (C_3). This procedure is repeated, continually expanding the training set and retraining the classifier with N queries (we set N=500 in our experiments). In this scenario, users' clicks are treated as ground truth and the classifier's goal is to rank clicked entities at position 1. We do not distinguish between clicks (e.g., satisfied clicks and non-

[3]We set $k = 20$ as it has shown a fair tradeoff between high recall (80.1% on our development set) and low computational expense.

Table 2: Performance of field similarity-based entity ranking methods using KB entity representations (KBER$_{sim}$) and dynamic collective entity representations (DCER$_{sim}$) in terms of MAP and P@1. Significance tested against KBER$_{sim}$. Oracle marks upper bound performance given our top-k scenario.

Run	MAP	P@1
KBER$_{sim}$	0.5579	0.4967
DCER$_{sim}$	**0.5971**$^{\blacktriangle}$	**0.5573**$^{\blacktriangle}$
Oracle	0.6653	0.6653

satisfied clicks), and we leave more advanced user models, e.g., which incorporate skips, as future work.

To show the robustness of our method we apply five-fold cross-validation over each run, i.e., we generate five alternatively ordered query logs by shuffling the queries. We keep the order of the dynamic description sources fixed to avoid conflating the effect of queries' order with that of the description sources.

Since we are interested in how our method behaves over time, we plot the MAP at each query over the five-folds, as opposed to averaging the scores over all chunks across folds (as in [7]) and losing this temporal dimension. In addition to reporting MAP, we report on P@1, as there is only a single relevant entity per query in our experimental setup. We test for statistical significance using a two-tailed paired t-test. Significant differences are marked $^{\blacktriangle}$ for $\alpha = 0.01$.

6. RESULTS AND ANALYSIS

We report on the experimental results for each of our three experiments, in turn, and provide an analysis of the results to better understand the behavior of our method.

6.1 Dynamic collective entity representations

In our first experiment, we explore the impact of the description sources we use for constructing dynamic collective entity representations. We compare the KBER$_{sim}$ baseline, which incorporates field similarity on KB descriptions, to our DCER$_{sim}$ method, which incorporates field similarity features on all description sources (web anchors, tweets, tags, and queries). Table 2 shows the performance in terms of MAP and P@1 of the baseline (KBER$_{sim}$) and DCER$_{sim}$ after observing all queries in our dataset. We include an oracle run as a performance upper bound given our top-k retrieval scenario.

The results show that the dynamic collective entity representations manage to significantly outperform the KB entity representations for both metrics, and that DCER$_{sim}$ presents the correct entity at the top of the ranking for over 55% of the queries.

Next, we look into the impact on performance of individual description sources for our dynamic collective entity representations. We add each source individually to the KBER$_{sim}$ baseline. Figure 2 shows how each individual description source contributes to more effective ranking, with KB+tags narrowly outperforming KB+web as the best single source. Combining all sources into one, yields the best results, outperforming KB+tags by more than 3%. We observe that after about 18,000 queries, KB+tags overtakes the (static) KB+web method, suggesting that newly incoming tags yield higher ranking effectiveness. All runs show an upward trend and seem to level out around the 30,000th query. This pattern is seen across ranking methods, which indicates that the upward trend can be attributed to the addition of more training data (queries).

Table 3 lists the results of all methods along with the improvement rate (relative improvement when going from 10,000 to all queries). The runs that incorporate dynamic description sources (i.e., KB+tags, KB+tweets, and KB+queries) show the highest relative improvements (at 6.4%, 6.1% and 6.9% respectively). Interestingly, for P@1, the KB+queries method yields a substantial relative improvement (+8.9%), indicating that the added queries provide a strong signal for ranking the clicked entities at the top.

The comparatively lower learning rates of methods that incorporate only static description sources (KBER$_{sim}$ and KB+web yield relative improvements of +5.8% and +5.5%, respectively), suggest that the entity content from dynamic description sources effectively contributes to higher ranking performance as the entity representations change and the ranker is able to reach a new optimum. DCER$_{sim}$, the method that incorporates all available description sources, shows a comparatively lower relative improvement, which is likely due to it hitting a ceiling, and not much relative improvement can be gained.

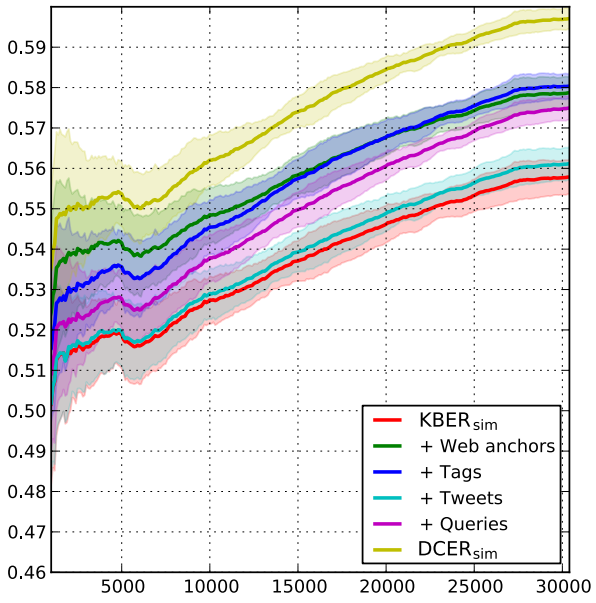

Figure 2: Impact on performance of individual description sources. MAP on the y-axis, number of queries on the x-axis. The line represents MAP, averaged over 5-folds. Standard deviation is shown as a shade around the line. This plot is best viewed in color.

Table 3: Comparison of relative improvement between runs with different field similarity features. We report on MAP and P@1 at query 10,000 and the last query. Rate corresponds to the percentage of improvement between the 10,000th and final query. Significance tested against KBER$_{sim}$.

Run	MAP (10k)	MAP (end)	Rate	P@1 (10k)	P@1 (end)	Rate
KBER$_{sim}$	0.5274	0.5579	+5.8%	0.4648	0.4967	+6.9%
KB+Web$_{sim}$	0.5485	0.5787$^{\blacktriangle}$	+5.5%	0.4965	0.5282$^{\blacktriangle}$	+6.4%
KB+Tags$_{sim}$	0.5455	0.5804$^{\blacktriangle}$	+6.4%	0.4930	0.5317$^{\blacktriangle}$	+7.8%
KB+Tweets$_{sim}$	0.5290	0.5612$^{\blacktriangle}$	+6.1%	0.4673	0.5021$^{\blacktriangle}$	+7.5%
KB+Queries$_{sim}$	0.5379	0.5750$^{\blacktriangle}$	**+6.9%**	0.4813	0.5242$^{\blacktriangle}$	**+8.9%**
DCER$_{sim}$	0.5620	**0.5971**$^{\blacktriangle}$	+6.2%	0.5178	**0.5573**$^{\blacktriangle}$	+7.6%

6.1.1 *Feature weights over time*

A unique property in our scenario is that, over time, more training data is added to the system, and more descriptions from dynamic sources come in, both of which are expected to improve system's performance. To better understand our system's behavior, we look into the learned feature weights for both a static system and our dynamic one at each retraining interval. Figure 3 shows the weights for six static fields (categories, title, anchors, links, text, redirects) and three dynamic fields (queries, tweets, tags).

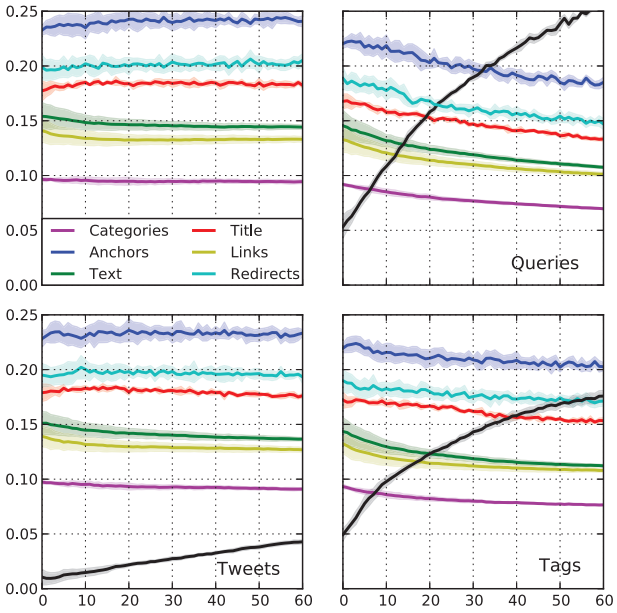

Figure 3: Feature weights over time: y-axis shows (relative) feature weights, x-axis shows each chunk of 500 queries where the ranker is retrained. Starting from top left in a clock-wise direction we show the following runs: KBER$_{sim}$ (baseline), KB+Queries, KB+Tags, KB+Tweets. The black line shows the dynamic description source's weight. This plot is best viewed in color.

The KBER$_{sim}$ baseline shows little change in the feature weights as more training data is added. The anchors' weight increases slightly, while the text and categories fields' weights steadily decline over time. The latter two features show a steadily decreasing standard deviation, indicating that the ranker becomes more confident in the assigned weights.

For KB+queries it is apparent that the queries field weight increases substantially as both the ranking method receives more training data and the entity representations receive more content from the queries description source. At the same time, anchors, redirects, and title field weights, which were assigned consistently high weights in KBER$_{sim}$, seem to pay for the increase of the importance of queries. Text, category, and links show similar patterns to the baseline run; steadily declining weights, converging in terms of a decreasing standard deviation.

The KB+tags run shows a similar pattern as KB+queries: we observe an increase over time of the weight assigned to the field that holds content from the tag description source, at the cost of original KB fields, resulting in improved ranking effectiveness. Looking at KB+tweets, however, we observe a different pattern. The tweets field starts out with a very low weight, and although the weight steadily increases, it remains low. The ranker here, too, becomes more confident on this source's weight, with the standard deviation dissolving over time. Looking at the higher performance of KB+tweets in comparison to the KBER$_{sim}$ baseline, together with the field's weight increasing over time, we can conclude that the tweets that are added to the entity representations over time provide added value.

6.2 Modeling field importance

In our second experiment, we turn to the contribution of the additional field and entity importance features. Table 4 lists the results of the best performing run with only field similarity features (DCER$_{sim}$), the KBER baseline that incorporates the full set of features, and our proposed DCER method which likewise incorporates the full set of features.

Table 4: Comparison of relative improvement between the KBER baseline with field importance features for KB fields, and our DCER method with these features for all fields. We also show the best performing run without field and entity importance features. Significance tested against KBER.

Run	MAP (10k)	MAP (end)	Rate	P@1 (10k)	P@1 (end)	Rate
Oracle	–	0.6653	–	–	0.6653	–
DCER$_{sim}$	0.5620	0.5971	+6.2%	0.5178	0.5573	+7.6%
KBER	0.5853	0.6129	+4.7%	0.5559	0.5831	+4.9%
DCER	0.5923	**0.6200**$^{\blacktriangle}$	+4.7%	0.5655	**0.5925**$^{\blacktriangle}$	+4.8%

The results show that modeling field and entity importance significantly improves effectiveness of both the KBER$_{sim}$ baseline and the DCER$_{sim}$ runs. After running through the entire dataset, the performance of DCER approaches that of the oracle run which also explains why the differences between the two approaches here is rather small: we are very close to the maximum achievable score.

Figure 4 shows the performance of the two approaches over time. The pattern is similar to the one in Section 6.1: both lines show a steady increase, while DCER maintains to be the best performing.

6.3 Ranker adaptivity

In our third experiment, we compare our adaptive ranker (DCER), which is continuously retrained, to a non-adaptive baseline (DCER$_{na}$), which is only trained once at the start and is not retrained.

Table 5: Comparing relative improvement between runs with and without an adaptive ranker. Statistical significance tested against their non-adaptive counterparts.

Run	Adaptive	MAP	P@1
KBER$_{na}$	no	0.5198	0.4392
KBER	yes	0.5579$^{\blacktriangle}$	0.4967$^{\blacktriangle}$
DCER$_{na}$	no	0.5872	0.5408
DCER	yes	**0.5971**$^{\blacktriangle}$	**0.5573**$^{\blacktriangle}$

Results in Table 5 show that incrementally re-training the ranker is beneficial to entity ranking effectiveness. For both KBER and DCER we see a substantial improvement when moving from one single batch training to continuously retraining the ranker. To better understand this behavior, we plot in Figure 5 the performance of all runs we consider over time and at the same time; Table 6 provides the detailed scores. Broadly speaking we observe similar patterns between adaptive and non-adaptive methods, and we identify three interesting points. First, for the non-adaptive methods, the absolute performance is substantially lower across the board. Second,

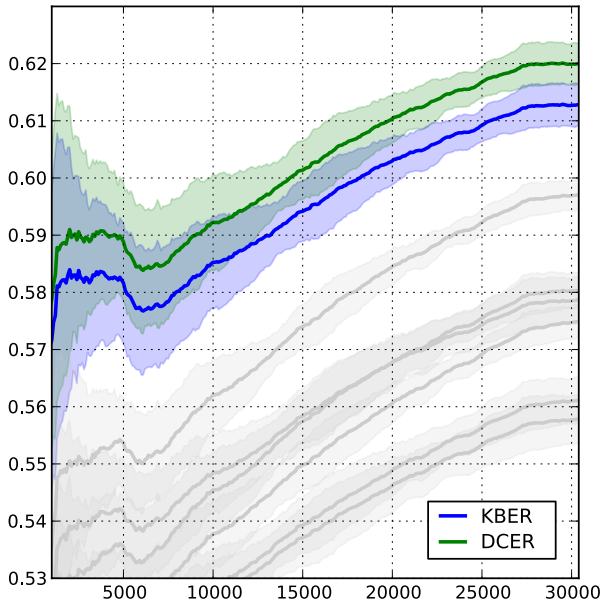

Figure 4: Runs with field similarity, field importance, and entity importance features. MAP on the y-axis, number of queries on the x-axis. The line represents MAP, averaged over 5-folds. Standard deviation is shown as a shade around the line. This plot is best viewed in color.

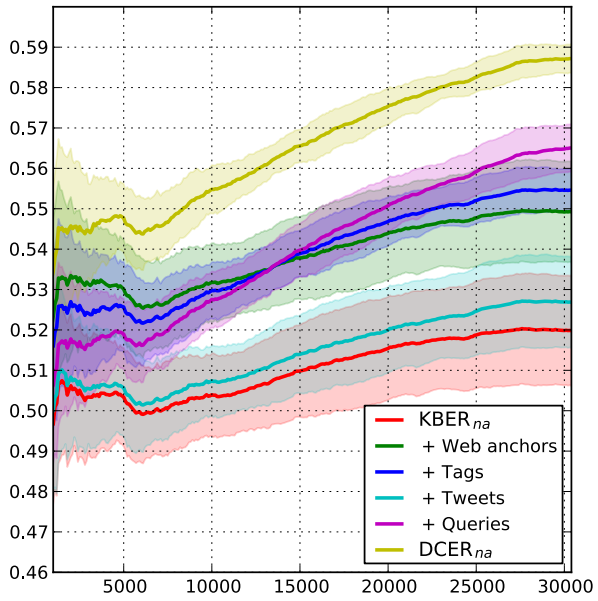

Figure 5: Individual description sources with nonadaptive ranker. MAP on the y-axis, number of queries on the x-axis. The line represents MAP, averaged over 5-folds. Standard deviation is shown as a shade around the line. This plot is best viewed in color.

for the adaptive methods, the standard deviation (as shown in Figures 2 and 5) is substantially lower, which indicates that retraining increases the ranking method's confidence in optimally combining the different descriptions for the entity representation. Third, the learning rates in the adaptive setting are substantially higher, reaffirming our observation that learning a single global feature weight vector is not optimal.

Table 6: Comparing relative improvement between nonadaptive runs. Significance tested against KBER$_{na}$.

Run	MAP (10k)	MAP (end)	Rate	P@1 (10k)	P@1 (end)	Rate
KBER$_{na}$	0.5040	0.5198	+3.1%	0.4286	0.4392	+2.5%
KB+Web$_{na}$	0.5318	0.5493$^{\blacktriangle}$	+3.3%	0.4698	0.4829$^{\blacktriangle}$	+2.8%
KB+Tags$_{na}$	0.5298	0.5546$^{\blacktriangle}$	+4.7%	0.4671	0.4904$^{\blacktriangle}$	+5.0%
KB+Tweets$_{na}$	0.5074	0.5269$^{\blacktriangle}$	+3.8%	0.4334	0.4490$^{\blacktriangle}$	+3.6%
KB+Queries$_{na}$	0.5275	0.5650$^{\blacktriangle}$	**+7.1%**	0.4659	0.5090$^{\blacktriangle}$	+9.2%
DCER$_{na}$	0.5548	**0.5872**$^{\blacktriangle}$	+5.8%	0.5063	0.5408$^{\blacktriangle}$	+6.8%

Table 6 shows that the difference in learning rates between methods that only incorporate static description sources (KBER, KB+web) and methods that incorporate dynamic sources is pronounced, in particular for the tags and queries sources. This indicates that even with fixed feature weights, the content that comes in from the dynamic description sources yields an improvement in entity ranking effectiveness. Finally, the methods that only incorporate static description sources also show lower learning rates than their adaptive counterparts, which indicates that retraining and adapting feature weights is desirable even with static entity representations.

7. CONCLUSIONS

In this paper we have presented a method for constructing dynamic collective entity representations by leveraging collective intelligence to improve entity ranking. Our results demonstrate that incorporating dynamic description sources into dynamic collective entity representations enables a better matching of users' queries to entities. Furthermore, we show how continuously updating the ranker leads to improved ranking effectiveness in dynamic collective entity representations.

Our study of the impact of dynamic collective entity representations was performed in a controlled scenario, where we collect and leverage different data sources. One restriction of working with these freely available resources is that it proves hard to find aligned and sizeable datasets. In particular, the temporal misalignment between different corpora prevents the analysis of temporal patterns that may span across sources (e.g., queries and tweets showing similar activity around entities when news events unfold). In part, these restrictions can be circumvented, e.g., increasing the (comparatively) low number of tweets by enriching them through e.g., entity linking methods [25], where entities identified in tweets could be expanded. Additional challenges and opportunities may arise when increasing the scale of the data collections. Opportunities may lie in exploiting session or user information for more effective use of user interaction signals. Challenges include so-called "swamping" [31] or "document vector saturation" [19], i.e., entity drift that is more prone to happen when the size of the data collections increase.

Acknowledgments. This research was supported by Amsterdam Data Science, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELIAS, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project nr 027.012.105, the Netherlands Insti-

tute for Sound and Vision, 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, 652.002.001, the Yahoo Faculty Research and Engagement Program, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] E. Amitay, A. Darlow, D. Konopnicki, and U. Weiss. Queries as anchors: selection by association. In *HYPERTEXT '05*, pages 193–201, 2005.

[2] K. Balog and K. Nørvåg. On the use of semantic knowledge bases for temporally-aware entity retrieval. In *Proceedings of ESAIR '12*, pages 1–2, 2012.

[3] K. Balog, M. Bron, and M. de Rijke. Category-based query modeling for entity search. In *ECIR '10*, pages 319–331, 2010.

[4] K. Balog, A. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. Overview of the TREC 2009 entity track. In *TREC '09*, 2010.

[5] K. Balog, P. Serdyukov, and A. de Vries. Overview of the TREC 2010 entity track. In *TREC '10*, 2011.

[6] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW '07*, pages 501–510, 2007.

[7] R. Bekkerman, A. McCallum, and G. Huang. Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. *Center for Intelligent Information Retrieval, Technical Report IR*, 418, 2004.

[8] B. Billerbeck, G. Demartini, C. Firan, T. Iofciu, and R. Krestel. Ranking entities using web search query logs. In *ECDL '10*, pages 273–281, 2010.

[9] A. Broder, E. Gabrilovich, V. Josifovski, G. Mavromatis, D. Metzler, and J. Wang. Exploiting site-level information to improve web search. In *CIKM '10*, pages 1393–1396, 2010.

[10] A. de Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *INEX '07*, pages 245–251, 2008.

[11] G. Demartini, A. de Vries, T. Iofciu, and J. Zhu. Overview of the INEX 2008 entity ranking track. In *INEX '08*, pages 243–252, 2009.

[12] G. Demartini, T. Iofciu, and A. de Vries. Overview of the INEX 2009 entity ranking track. In *INEX '09*, pages 254–264, 2010.

[13] M. Efron, P. Organisciak, and K. Fenlon. Improving retrieval of short texts through document expansion. In *SIGIR '12*, pages 911–920, 2012.

[14] N. Eiron and K. S. McCurley. Analysis of anchor text for web search. In *SIGIR '03*, pages 459–460, 2003.

[15] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie. Smoothing clickthrough data for web search ranking. In *SIGIR '09*, pages 355–362, 2009.

[16] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*, pages 249–258, 2011.

[17] K. Hong, P. Pei, Y.-Y. Wang, and D. Hakkani-Tur. Entity ranking for descriptive queries. In *SLT '14*, 2014.

[18] R. Kaptein, P. Serdyukov, A. de Vries, and J. Kamps. Entity ranking using wikipedia as a pivot. In *CIKM '10*, pages 69–78, 2010.

[19] C. Kemp and K. Ramamohanarao. Long-term learning for web search engines. In *PKDD '02*, pages 263–274, 2002.

[20] R. Kumar and A. Tomkins. A characterization of online browsing behavior. In *WWW '10*, pages 561–570, 2010.

[21] C.-J. Lee and W. B. Croft. Incorporating social anchors for ad hoc retrieval. In *OAIR '13*, pages 181–188, 2013.

[22] F. Li, M. L. Lee, and W. Hsu. Entity profiling with varying source reliabilities. In *KDD '14*, pages 1146–1155, 2014.

[23] T.-Y. Liu. Learning to rank for information retrieval. *FnTIR*, 3(3):225–331, 2009.

[24] C. Macdonald, R. L. Santos, I. Ounis, and B. He. About learning models with multiple query-dependent features. *ACM TOIS*, 31(3):11:1–11:39, 2013.

[25] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM '12*, pages 563–572, 2012.

[26] D. Metzler, J. Novak, H. Cui, and S. Reddy. Building enriched document representations using aggregated anchor text. In *SIGIR '09*, pages 219–226, 2009.

[27] G. Mishne and J. Lin. Twanchor text: A preliminary study of the value of tweets as anchor text. In *SIGIR '12*, pages 1159–1160, 2012.

[28] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *JMLR Workshop and Conference Proceedings*, 14:77–89, 2011.

[29] D. Mottin, T. Palpanas, and Y. Velegrakis. Entity ranking using click-log information. *Intelligent Data Analysis*, 17(5): 837–856, 2013.

[30] M. Noll and C. Meinel. The metadata triumvirate: Social annotations, anchor texts and search queries. In *International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT '08)*, pages 640–647, 2008.

[31] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04*, pages 42–49, 2004.

[32] F. Scholer, H. E. Williams, and A. Turpin. Query association surrogates for web search: Research articles. *JASIST*, 55(7): 637–650, 2004.

[33] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *TKDE*, 27(2):443–460, 2014.

[34] A. Singhal and F. Pereira. Document expansion for speech retrieval. In *SIGIR '99*, pages 34–41, 1999.

[35] K. M. Svore and C. J. Burges. A machine learning approach for improved bm25 retrieval. In *CIKM '09*, pages 1811–1814, 2009.

[36] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *CACM*, 53(8):80–88, 2010.

[37] M. Tsagkias, W. Weerkamp, and M. de Rijke. News comments: Exploring, modeling, and online prediction. In *ECIR '10*, pages 191–203, 2010.

[38] A.-M. Vercoustre, J. A. Thom, and J. Pehcevski. Entity ranking in wikipedia. In *SAC '08*, pages 1101–1106, 2008.

[39] T. Westerveld, W. Kraaij, and D. Hiemstra. Retrieving web pages using content, links, urls and anchors. In *TREC '01*, 2001.

[40] M. Wu, D. Hawking, A. Turpin, and F. Scholer. Using anchor text for homepage and topic distillation search tasks. *JASIST*, 63(6):1235–1255, 2012.

[41] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM '04*, pages 118–126, 2004.

[42] H. Zaragoza, N. Craswell, M. Taylor, S. Saria, and S. Robertson. Microsoft Cambridge at TREC-13: Web and hard tracks. In *TREC '04*, 2004.