Related Event Discovery

Cheng Li[1]*, Michael Bendersky[2], Vijay Garg[2], Sujith Ravi[2]
[1]School of Information, University of Michigan, Ann Arbor, MI, USA
[2]Google, Inc., Mountain View, CA, USA
lichengz@umich.edu, {bemike, vijaygarg, sravi}@google.com

ABSTRACT

We consider the problem of discovering local events on the web, where events are entities extracted from webpages. Examples of such local events include small venue concerts, farmers markets, sports activities, etc. Given an event entity, we propose a graph-based framework for retrieving a ranked list of related events that a user is likely to be interested in attending. Due to the difficulty of obtaining ground-truth labels for event entities, which are temporal and are constrained by location, our retrieval framework is unsupervised, and its graph-based formulation addresses (a) the challenge of feature sparseness and noisiness, and (b) the semantic mismatch problem in a self-contained and principled manner.

To validate our methods, we collect human annotations and conduct a comprehensive empirical study, analyzing the performance of our methods with regard to relevance, recall, and diversity. This study shows that our graph-based framework is significantly better than any individual feature source, and can be further improved with minimal supervision.

Keywords

event discovery; event retrieval; Schema.org entities; web events

1. INTRODUCTION

Recently, the problem of recommending events to users has received considerable attention from the research community. Given a set of scheduled events such as concerts, hiking activities, or conferences, a number of algorithms were proposed to recommend events to attend. These algorithms either focus on events in a particular domain such as scientific talks [26], or events from the Event-Based Social Networks (EBSN), such as Meetup and Plancast [11, 25]. Since it is relatively easy to collect user feedback and interaction information when there are only a few well-curated event sources available, existing methods require supervision when tackling the event recommendation task.

In this paper we address the task of discovering related events in a much more general setting and at a larger scale – we consider all available events extracted from pages across the entire web. More specifically, we only require an extracted event to be described by three fields: *What*, *Where*, and *When*, corresponding respectively to event title (a short description of the event), location and time. Given such an event, our task is to find a ranked list of events related to it. Event *A* is related to event *B* if a user, attending or interested in event *A*, is also interested in attending event *B*.

*This work was done while the author was at Google, Inc.

WSDM 2017 February 06-10, 2017, Cambridge, United Kingdom
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4675-7/17/02.
DOI: http://dx.doi.org/10.1145/3018661.3018712

In this paper, we study events annotated by Schema.org[1] markup, which naturally provides the three fields. We use Schema.org since it is a rich source of structured annotations – recent research shows that more than 30% of web pages have Schema.org markup, with *Events* being one of the most popular categories [18]. However, it is important to note that our methodology is not limited to Schema.org events. Augmented with techniques to automatically extract events from any free-text web pages, e.g., [16], our system could become a powerful tool for related event discovery across the web.

It is important to point out that the process of event discovery and recommendation is exploratory, rather than precision-driven, as users often do not have a specific information need in mind, and are more interested in exploring related nearby events. Thus, presenting hiking events to a user who attended a hiking event in the past might be safe, but does not provide a serendipitous experience. Instead, our aim is to retrieve and recommend more diverse semantically related events such as rock climbing, caving or backpacking. Table 1 demonstrates this point by showing related events retrieved by our system for two example query events.

Table 1: Examples of related events, retrieved by our system in response to example query events.

Query event	*Hiking Trip: Armstrong Woods With Marcia 7-8 Miles.*
Relevant events	Rock Climbing: Yosemite: Snake Dike.
	Caving: Church Cave, Fresno County, Kings Canyon National Park.
	Sunol Wilderness Backpacking Overnight! 3.74Mi 1400Ft.
Query event	*Free Energy System Tour.*
Relevant events	Healing Wednesdays – Get relief from your emotional and physical pains.
	Isha Kriya Guided Meditation – Free Class.
	Yoga and Yoga-Nidra.

In addition, since event relevance is restricted by location and time (past events or events in other countries or states are not likely to be relevant), the inventory of potentially related events can be restricted. Therefore, simple retrieval methods based solely on event title or body matching will suffer from low recall problem.

Retrieving semantically relevant events could alleviate the recall problem, and possibly lead to serendipitous discoveries. In order

[1]http://schema.org/Event.

to address semantics, one might argue that important keywords in event descriptions, such as *hiking* or *climbing*, could be looked up in a knowledge base to find semantically close terms. However, it is not always easy to identify such keywords or phrases in a sentence. Even if we have successfully extracted the keywords, given the diversity of events, corresponding answers might not exist in a knowledge base. For example, the second query in Table 1 shows an event about human body energy and healing. Simply searching keywords *free energy system* in a knowledge base will yield results about thermodynamic systems, while in reality, users are much more likely to be interested in attending events featuring healing, meditation or yoga.

Therefore, in this paper we investigate the feasibility of tackling the semantic matching challenge and alleviating the retrieval recall problem via a more general and formal approach. We propose a self-contained graph-based framework for retrieving a ranked list of related events. This framework takes into account multiple contextual features that can be associated with a web-extracted event, including extracted field text, surrounding text of the page, taxonomy classification, related queries, etc. The features are integrated into a single event-feature bipartite graph, and retrieval is done via graph propagation methods.

Compared with prior work, the detailed contributions of our work are as follows. First, we formulate the problem of local event recommendation, and propose a novel graph-based retrieval framework to solve this problem. Unlike the existing methods, our framework scales the solution beyond a few curated sources of information to the open web, and does not require ground-truth data, which can be difficult to obtain for events, due to their transient and local relevance. However, we do demonstrate that our framework permits integration of labeled examples via learning-to-rank.

Second, we propose a practical solution for extracting event-related features from sparse and noisy data. Events, due to their transient and local nature, do not share many of the clean and well-formatted properties enjoyed by other common entities, e.g., people or organizations. These common entities are usually annotated by RDF with rich information recorded in knowledge bases, and are mentioned multiple times in different webpages, providing sufficient context information. In contrast, events are described by a short title, typically not stored in a knowledge base, and are present only at one, or at most a handful of web pages. Moreover, these pages often do not give details about the events, other than a short title and description. Given these challenges, we resort to a variety of contexts of the event on the web page to extract various signals, enriching the limited information obtained from the fields of the event entity. However, these signals should be used with care, as they are noisy and sparse. To this end, we provide a description of a data pipeline to (a) mine data from different sources associated with the event page, (b) aggregate data from these different sources into a unified feature representation, and (c) remove noisy / irrelevant features through a novel *local stopword detection* technique.

Finally, we provide a comprehensive empirical study of the proposed methods. We collect human annotations and analyze the performance of our methods in terms of relevance, recall, and diversity for event recommendation in four large US metropolitan areas. Our study shows that the graph-based framework is significantly better than any individual feature source. It also outperforms standard retrieval baselines, including title matching, rank fusion and pseudo-relevance feedback, and can be further improved with minimal supervision.

While the focus of our paper is on events, our proposed methods are general and can be applied to other long-tail entities that are constrained by time or location. A case in point are local business entities (e.g., restaurants, small retailers, etc.), for which not enough (or not at all) rating information exists online.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 provides a description of our dataset. Section 4 proposes methods to tackle the challenges, followed by the setup of experiments in Section 5. Section 6 presents the results of experiments, including a detailed analysis of methods and features. We conclude the paper and discuss future work in Section 7.

2. RELATED WORK

A survey of literature shows that existing work studies events from particular domains or websites. In this work, we consider any events that could be extracted from the web into three fields – *What*, *Where*, and *When*. In the absence of user information, we attempt to employ unsupervised learning techniques to retrieve related events.

2.1 Event recommendation

Some studies focus on recommending events of a particular category. Minkov et al. [26] adopt collaborative filtering methods to recommend scientific seminars to users in universities based on feedback. Collecting user feedback from a Belgian cultural event website, popular recommendation approaches are examined in [15].

An active line of research is event recommendation on the Event-Based Social Networks (EBSN), such as Meetup and Plancast. Both content based and collaborative filtering methods are studied. Researchers try to mine information from co-participating links, follow links, and user profiles [22, 24, 23, 29, 25].

The above methods study events from a particular domain, or from a few event websites, where user information is available for supervised learning. In contrast, we consider a more general setting where events are extracted from webpages. Considering the transient and local relevance of events, ground-truth information is hard to come by. Therefore, we exploit retrieval-based and unsupervised learning techniques to discover related events.

2.2 Entity search

Given that events are Schema.org annotated entities, a line of research broadly related to this work is entity search. Entity search tasks have received attention from both TREC and INEX conferences, where various tasks have been proposed [12, 2]. In these tasks, entities are defined by their homepages (e.g., a Wikipedia page describing the entity), which provides rich information for retrieval. First, category and link information is available in knowledge bases [12], facilitating the interpretation of the relationships between entities [1, 5]. Second, each homepage provides abundant text information to describe the entity. Utilizing both text and category information, performance can be improved by language modeling [40, 42]. Graus et al. [17] dynamically incorporate various sources for entity representation.

Another branch of related work comes from the community of semantic web, where RDF is usually used to describe entities. RDF datasets are graphs where nodes are resources and edges are relations between resources. PageRank can be used to rank RDF graphs [14, 19]. Ranks from RDF graphs can be complemented by structures of source webpages and knowledge bases [20, 13].

Studies most relevant to ours formulate entity search as a retrieval task. Conrad and Utt [7] create pseudo documents by collapsing paragraphs mentioning an entity. Language models can be built on words surrounding the entities [27, 31]. For RDF objects, attributes are treated as words in documents for ranking [28, 6].

When relevance judgments or user clicks are available, learning to rank techniques can be employed. Dali et al. [9] rank RDF entities by extracting features from RDF graphs and external knowledge source. Ranking entities in web search has been studied by extracting statistical features, or graph based features from query logs, Flickr, tweets and structured collections [38, 21].

A relevant task is TREC Enterprise search, which aims to find experts pertaining to a query [35].

The above methods make at least one of the following assumptions for entities: (1) associated with a homepage; (2) expressed by RDF; (3) defined in knowledge bases; (4) popular enough to appear in multiple webpages. In contrast, we address events with following characteristics:

- There are usually no homepages to fully describe the events. Though there is a webpage displaying the event details, surrounding context might be noisy or irrelevant.
- Unlike RDF entities, no explicit links between events exist, and descriptions for these events are often terse and ambiguous.
- New events are created on a daily basis, making it hard to define and store them in knowledge bases.
- Usually each event only occurs once in one webpage. As a result, it would be questionable to utilize occurrence information and aggregated surrounding context information.

These challenges complicate the task of discovering related events, making it infeasible for the application of most methods proposed for standard entity search scenarios. These difficulties also necessitate the investigation of how approaches requiring minimal supervision can be adapted to tackle the present problem.

3. DATASET

As noted above, event extraction from free-text webpages is not the focus of present work. For simplicity, we utilize Schema.org annotated events, which are described by three fields: *What*, *Where*, and *When* [16], respectively corresponding to event title (a short description of the event), location and time. These events are extracted from a large web crawl of English webpages, resulting in a total of 6,105,223 events. Although in this work we use a proprietary corpus, similar datasets can be obtained by extracting events annotated by Schema.org from public corpora like ClueWeb12[2] or Common Crawl[3] (see, e.g., [16] for more details). Additionally, Schema.org is an open-source mark-up that can be parsed by anyone.

Events are typically locality-sensitive – people usually only care about events in their geographical proximity. Hence, it is reasonable to consider relevance of events within a certain range. For evaluation, in experiments we study four major metropolitan regions in the United States with a significantly large inventory of extracted events. We focus on the United States due to the prevalence of Schema.org events in this country [16]. However, our proposed framework does not utilizes knowledge specific to United States and is general enough to be applied to other areas.

After the four regions have been selected, we set a geo-point at the center of each region, and events in a radius of 100 miles from this geo-point are included. Table 2 describes statistics of these regions.

4. EVENT DISCOVERY METHODS

In this section, we formulate a general framework for event discovery on the web. Our framework is retrieval-based and unsupervised, since event entities extracted from the web will, in general, have no ground-truth labels available. While the main focus of this paper is on events, the proposed framework is general enough to be applied to discover other entity types, as it only requires the following two conditions to be met:

1. Entities (e.g., events) can be reliably extracted from web pages.
2. We can extract some features describing the entity from its context (e.g., text describing the entity, its surrounding text, etc.).

We start our discussion in Section 4.1, where we first describe how event discovery satisfies these conditions, and provide the general framework formulation. Then, in the remainder of this section, we discuss a concrete implementation of this framework for event discovery. Please note that since this is the first attempt to tackle event retrieval at such a large scale, our study is exploratory and does not address the efficiency issues related to online event retrieval. We discuss the adaptation of our methods to efficient online algorithms in Section 6.5.

[2]http://lemurproject.org/clueweb12/

[3]http://commoncrawl.org/

Table 2: Statistics of extracted events for four metropolitan regions in the United States.

Region	# events	Geo-point
New York	57,780	40.66, -73.93
San Francisco Bay Area	28,442	37.40, -122.10
Chicago	23,873	41.84, -87.68
Philadelphia	57,395	40.01, -75.13

4.1 Problem formulation

We start our discussion by assuming that we can extract events (or other entity types) from web pages. As described in Section 3, Schema.org markup provides an extensive source of such extractions, and provides information on multiple types of entities.

Once extracted, each event is associated with a set of features based on its text, or other context that can be associated with it. Due to the noisiness and sparseness of such features, they may be unreliable for direct use, and could have low precision and recall.

For instance, let us consider simply matching events by their title. This approach may lead to many term mismatches, due to the short title length. On one hand, matching by title terms would fail to find the connection between *hiking* and *climbing*, which are both outdoor sports. On the other hand, it might suggest that *Soccer Night on Friday* and *Movie Night on Friday* are related, as they share most terms. To retrieve events more reliably, we propose two approaches to aggregating multiple feature categories.

4.1.1 Event Search

Assuming a list of k feature categories, we denote $\mathcal{F}_S^j$ a set of features extracted from feature category j associated with a source (or query) event S. Similarly, $\mathcal{F}_T^j$ is a set of features extracted from feature category j for some target event T. A simple approach is combining the similarity of these features into a single aggregate similarity score $sc(S,T)$, which can be generally expressed as:

$$sc(S,T) = \phi(sc(\mathcal{F}_S^1, \mathcal{F}_T^1), \ldots, sc(\mathcal{F}_S^k, \mathcal{F}_T^k)), \qquad (1)$$

where ϕ is some aggregation function. This approach is akin to the *rank fusion* method, which was proved to be successful in ad hoc information retrieval [8].

We refer to this aggregation approach **COMBINER**. Its concrete implementation is described in more detail in Section 4.3.

An important shortcoming of COMBINER is that it does not address semantics sufficiently, as each source-target event pair is examined independently of other events. If we leverage the connections between events, more semantic information could potentially be exploited. While, in principle, it is possible to mine external knowledge to acquire such connections, it is not easy to extract the right keywords from event titles, and map them correctly to related terms in a semantic knowledge base, as explained in Section 1. Hence, we propose a more self-contained approach instead.

We build a feature-event bipartite graph, where event similarity can be propagated through the edges. Intuitively, in this framework two events may be similar if they share connections to other similar events. Formally, this approach can be described as follows:

$$\begin{aligned} sc(S,T) = \phi(&sc(\mathcal{F}_S^1, \mathcal{F}_T^1, \bigcup_{N \in \mathcal{N}_{S,T}} \mathcal{F}_N^1), \ldots, \\ &sc(\mathcal{F}_S^k, \mathcal{F}_T^k, \bigcup_{N \in \mathcal{N}_{S,T}} \mathcal{F}_N^k)), \end{aligned} \quad (2)$$

where $\mathcal{N}_{S,T}$ is the neighborhood of the events S and T in the feature-event graph. Note that $\mathcal{N}_{S,T}$ models the entire graph structure to find related events, not just the immediate neighborhood.

We call this graph-based approach **EXPANDER**, as it expands the set of compared results by their graph neighborhoods. Its concrete implementation is described in more detail in Section 4.4.

4.2 Feature preparation

We give an overview of the feature processing pipeline that extracts the features, normalizes them, and reduces the noise in the processed data. The resulting features are fed into either COMBINER or EXPANDER described in the previous section for similarity computations. This general process is illustrated in Figure 1.

4.2.1 Feature extraction

A simple method of retrieving similar events is pure textual matching of event descriptions. However, event descriptions are short and term mismatch is very common. E.g., *Miles Davis* and *John Coltrane* are both jazz musicians, but textual matching will not be able to reveal this fact. As a result, this simple method could suffer from limited accuracy and low recall. However, information from the event entity itself is limited, as an event is only described by three fields: *What*, *Where*, and *When*.

Therefore, to enrich the information obtained from events, we resort to their context. The context, in its most general sense, can be represented by a set of feature categories associated with an event entity, which are extracted from the source webpage(s) displaying the event. Specifically, we consider these feature categories:

1. TITLE. The title, or *What* field of the event entity, which usually describes the event in one to two sentences.
2. SURROUND. Surrounding text around the event entity, for which the window size is set to 2,000 characters.
3. QUERY. We use a query log from a commercial search engine to mine query information. A query will be collected from the log if it leads to a click on the source webpage containing an event. These queries are general and might not be event-specific.
4. ANCHOR. Anchor text pointing to the source webpage(s).
5. TAXONOMY. Taxonomic categorization of the source webpage(s). We use a proprietary classifier, trained to output the taxonomic categories of each webpage, using a predefined hierarchical tree. This tree is almost the same as the hierarchical ontology scheme provided by the Open Directory Project[4] – a few top categories, e.g., arts, business, and computers, are expanded to more specific ones like painting, advertising and computer memory. A webpage could be labeled by multiple categories, including both abstract and specific ones, but we only use leaf categories in the tree as features, preventing TAXONOMY based ranker from retrieving events that are only remotely related.

The first four categories of information are textual, and are stemmed by Porter stemmer, and converted to unigram and bigram features.

[4]http://www.dmoz.org/

Events are de-duplicated after feature extraction, making it easier to merge feature values of duplicate events. For simplicity, events with identical titles are considered as duplicates. To merge features of duplicate events, feature values of SURROUND, QUERY, and ANCHOR are summed up, and max is taken for TAXONOMY.

4.2.2 Noise reduction

Stopword removal and term reweighting are standard practice to reduce feature noise in information retrieval. Features with high frequency are considered as stopwords in this setting. One interesting finding is that even after stopword removal on the entire event set, frequent but non-important features remain when we focus on events of a specific region. For example, many events from New York contain the phrase *New York*, which might not be a stopword feature globally. To solve this problem, we propose the idea of *local stopwords*, which are frequent features for events of a particular region. This idea leads to two levels of stopword removal – global and local.

To reweight features, we apply log normalized TF-IDF. Feature frequency is logarithmically scaled to further scale down the frequent features, as we observe a rather skewed distribution of feature frequency, even after stopword removal. More specifically, $weight_{ij}$ of a $feature_j$ in a particular $event_i$ is calculated as:

$$weight_{ij} = \log(1 + tf_{ij}) \times \log(\frac{N}{df_j}) \quad (3)$$

where tf_{ij} is the frequency of $feature_j$ in $event_i$, N is the collection size, or in this context the total number of events, and df_j is the number of events where $feature_j$ appears.

4.3 Combiner

Representing events by a feature set allows us to obtain related events by measuring similarity between features. To instantiate Equation 1, we use Jeffreys-Kullback-Leibler divergence, which is a symmetric version of Kullback-Leibler divergence. We employ a measure on probability distributions because we treat each event as a distribution of features. We prefer the symmetry of this measure since we consider the similarity between events to be symmetric. The similarity score is normalized, with 0 being least similar and 1 being most similar. More precisely, given features $\mathcal{F}_S^j$ and $\mathcal{F}_T^j$, the similarity between source event S and target event T is defined as

$$sc(\mathcal{F}_S^j, \mathcal{F}_T^j) = \exp\left[-\frac{1}{2}\left(D(\mathcal{F}_S^j||\mathcal{F}_T^j) + D(\mathcal{F}_T^j||\mathcal{F}_S^j)\right)\right] \quad (4)$$

where $D(\cdot||\cdot)$ is the Kullback-Leibler divergence.

Given a source event, each individual ranker $ranker_j$ ranks target events based on one feature category j using score calculated in Equation 4. For example, QUERY-ranker computes the similarity score based on terms in queries that can be associated with events and ranks target events accordingly. As another example, using TITLE-ranker alone will result in a standard retrieval based on event titles.

Individual rankers could be unstable, and fail completely when the feature category is absent for one event. To combat this instability, we combine rankings from individual rankers into a single ranking list by Reciprocal Rank Fusion [8], which has empirically proven to be effective in combining individual rankers. Specifically, suppose $ranker_j$ assigns event T a ranking of $r_{S,T}^j$ w.r.t. some source event S. Then, the fused ranking score $sc(S,T)$ can be expressed as

$$sc(S,T) = \sum_j \frac{1}{r_{S,T}^j + K} \quad (5)$$

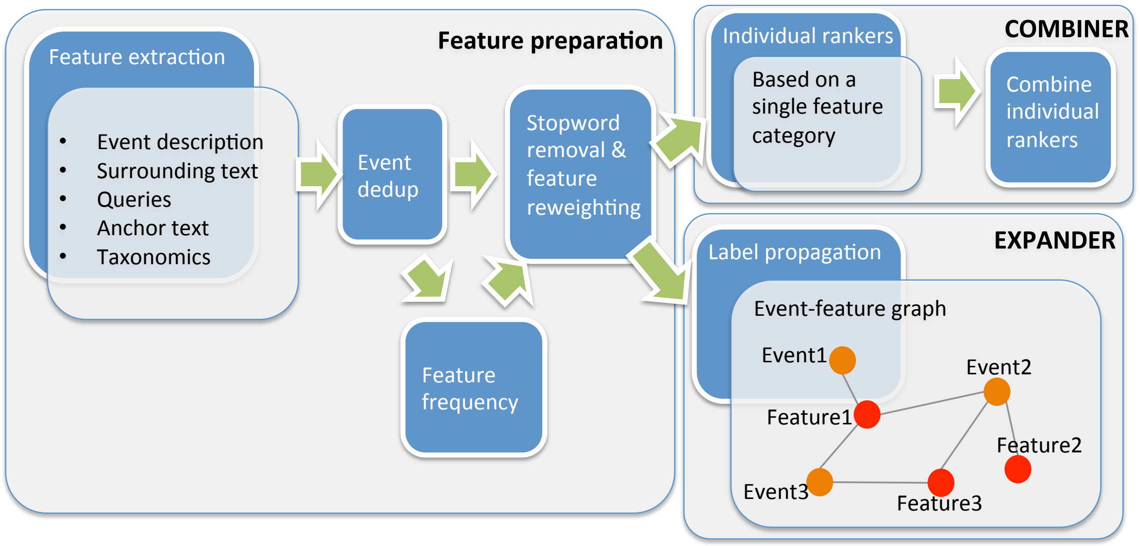

Figure 1: Related event discovery pipeline.

where K is a large predefined constant that reduces the impact of high rankings given by outlier rankers.

To illustrate the idea of COMBINER, consider a simple example using only TITLE- and QUERY-rankers. Suppose we have a source event S and a target event T. Our objective is to rank T by the similarity to S. Suppose TITLE ranks T at position 5, i.e., $r^{\text{TITLE}}_{S,T} = 5$, while QUERY ranks it at first position: $r^{\text{QUERY}}_{S,T} = 1$. In this case, COMBINER gives the similarity score between S and T as $sc(S,T) = \frac{1}{5+K} + \frac{1}{1+K}$.

4.4 Expander

COMBINER relies on individual rankers that make comparisons between each pair of events in a geographical region, which is computationally expensive. Furthermore, it might miss many semantically related events. Admittedly, QUERY- and TAXONOMY-rankers might help with semantics, but the information is limited and many events are not associated with any queries, or predefined taxonomic categories. To better address semantics, we use the following intuition: if event A is similar to event B, some of their features might be semantically related. This leads to the idea of constructing a graph in which (event and feature) data is represented by nodes while edges connect nodes that are related to each other. Edge weights are defined using a similarity function on node pairs and govern the strength of the semantical relationships between the nodes. To learn over this graph, we exploit the idea of label propagation [43, 4, 3, 37, 39, 32], a graph-based learning method that uses the information associated with each labeled "seed" node, and propagates these labels over the graph in a principled, iterative manner. This method scales well to large data size, which is an important aspect for our task.

4.4.1 Graph construction

There are two types of input sources for label propagation: the graph and the seed labels. The algorithm propagates the seed labels based on the provided graph structure, outputting a distribution of seed labels for each node in the graph.

Traditionally, label propagation is performed on homogeneous graphs, e.g., a graph only with events as nodes. However, this graph structure leads to some issues. First, absence of features in the graph results in a loss of information of the semantic connections between features. Second, edge weight, or similarity between events, has to be pre-defined by a rigid scoring function (similarly to COMBINER). In addition, edge weight cut-offs have to be pre-determined to control the graph density (in a fully dense graph, computing similarity between every pair of events will be prohibitively expensive).

To address these issues, we build an event-feature bipartite graph for label propagation, where both events and features are nodes. In this case, EXPANDER can naturally use feature information to find similar events, without defining in advance the similarity measure. An $event_i$ is linked to $feature_j$ if $event_i$ has $feature_j$, with edge weight defined in Equation 3. If a feature is shared by multiple feature categories, we collapse them. For example, "movie" from QUERY and TITLE are collapsed into one feature, with values added up. This practice reduces feature dimension, mitigating feature sparsity issues.

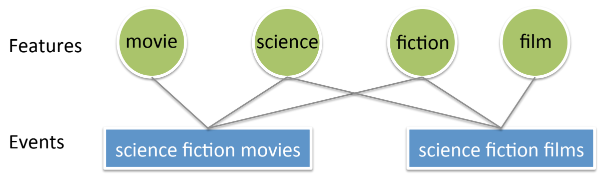

Figure 2: An example of event-feature bipartite graph for EXPANDER. Events, represented by blue rectangles, are connected to their features, which are in green circles.

An example of the constructed graph is shown in Figure 2. Though *movie* and *film* are not directly connected in Figure 2, sharing a large number of neighbors could promote their similarity.

Our constructed graph is very flexible – direct links between events or features could be added, if we had prior knowledge about their relationships. For instance, we could know from a thesaurus that feature *film* is a synonym for *movie*, and thus can be directly connected in the graph. In this exploratory study, we do not consider these direct links since we want to evaluate EXPANDER without additional knowledge, making it comparable to COMBINER.

Seed labels are ground-truth labels to be propagated across the graph. For example, if we know that event A and B are truly relevant, event A could have a label "event B". Unfortunately, we do not have such ground truth information available. As a result, we use identity labels as seeds, meaning that each event is a relevant label for itself. For example, event A is labeled as "event A".

4.4.2 *Objective function*

Given a graph constructed with event and feature nodes, the objective function of EXPANDER (a variant of [4]) simultaneously minimizes the following over all nodes in the graph:

- squared loss between true and induced label distribution.
- regularization term that penalizes neighboring feature nodes that have different label distributions from this event node.
- regularization term that smooths the induced label distribution towards the prior distribution, which is usually a uniform distribution in practice.

More precisely, given a single event node i with its feature neighbors $\mathcal{N}(i)$, the objective function being minimized is:

$$||\hat{Y}_i - Y_i||^2 + \mu_{np} \sum_{j \in \mathcal{N}(i)} w_{ij} ||\hat{Y}_i - \hat{Y}_j||^2 + \mu_{pp} ||\hat{Y}_i - U||^2 \quad (6)$$

where $\hat{Y}_i$ is the learned label distribution for event node i, Y_i is the true label distribution, μ_{np} is a predefined penalty for neighboring nodes with divergent label distributions, $\hat{Y}_j$ is the learned label distribution for feature neighbor j, w_{ij} is the weight of $feature_j$ in $event_i$, μ_{pp} is the penalty for label distribution deviating from the prior, a uniform distribution U.

The objective function for a feature node is alike, except that there is no first term, as there are no seed labels for feature nodes:

$$\mu_{np} \sum_{i \in \mathcal{N}(j)} w_{ij} ||\hat{Y}_j - \hat{Y}_i||^2 + \mu_{pp} ||\hat{Y}_j - U||^2 \quad (7)$$

It is important to note the connection between EXPANDER and iterative query expansion [33], as they both utilize neighbor's neighbors to find events. However, iterative query expansion requires users to specify the similarity metric between events, and the expansion process is ad hoc. In contrast, EXPANDER integrates multiple feature categories in a principled way, and labels are computed by optimizing an objective function over the entire graph.

5. EXPERIMENTAL SETUP

5.1 Parameters

For all parameters in the methods, we set them to values commonly used in practice. Specifically, K in Reciprocal Rank Fusion is set to 60, following [8]. To remove stopword features, we calculate frequency histogram for each feature category, and generally consider features within the 10th percentile to be stopwords. Label propagation is run for 5 iterations, with neighbor penalty μ_{np} set to 0.5, and prior penalty μ_{pp} 0.001.

5.2 Data preparation

In order to prepare data for human judgments, we randomly sampled 50 events as query events from each of the 4 regions listed in Section 3. To avoid bias, we would like to always present 20 candidate events to annotators for every query event. A simple method would be to take the top 10 candidates from each of the ranking lists from COMBINER and EXPANDER. However, some proposed candidates are shared by both methods, making the resultant list less than 20 events. To solve this problem, we adopted a process similar to interleaving [30]. Each time, we randomly picked one of the methods (COMBINER or EXPANDER), and took the candidate event with the highest score given by this method. This process continued until 20 candidates were fetched. The positions of these candidates were then shuffled before human annotations, mitigating position bias. In total, the collection process leads to 200 source (query) events and 4,000 target candidate events.

Before conducting the comprehensive evaluation, we first ran a preliminary study. In this pilot study, we found that annotators were attracted to candidate events with keyword matches in title, ignoring semantically relevant events. Therefore in the following studies, we remove candidates matching keywords with the query event (excluding stopword features). This helps us evaluate methods with respect to their semantic matching performance.

5.3 Collecting judgments

We use Amazon Mechanical Turk (AMT)[5], a popular crowdsourcing platform to collect human judgments. For each individual annotation task, which is called HIT in AMT, we show a query event and a list of 10 candidate events, together with the links to their source webpages. Note that a query event appears in two HITs, as there are 20 candidates per query. Annotators are asked to select the degree of similarity between the query event and each candidate event, with the following instructions

> *Two events are considered similar if someone who is interested in attending one of the events will also be interested in attending the other, regardless of their time or location.*

Note that we explicitly request the annotators to ignore location and time, as the event location is already accounted for during candidate generation, and restriction by date could significantly reduce the available event inventory. In real scenarios, time constraint can be easily met by adding a filter.

Each candidate event is rated by three levels of similarity: *similar*, *somehow similar*, and *not similar*. To guarantee the annotation quality, we specify the following requirements when hiring annotators: each task should be judged by 3 annotators, annotators have HIT approval rate no less than 80%, and they have at least 50 HITs approved. A payment of \$0.05 is made for each HIT per annotator. The agreement among annotators is 0.69. The relative low score suggests the diverse aspects of event relevance.

5.4 Metrics

We consider three metrics for evaluation, with each of them focusing on different aspects of the evaluation.

Prec@10-vote computes precision at 10 and defines ground-truth relevance by majority vote. That is, a candidate event is considered similar to a query event if at least 2 of the 3 annotators label it as *somehow similar* or *similar*.

Prec@10-relax is more relaxed, which considers an event as similar as long as a single annotator thinks so. We include this metric because event relevance could have diverse aspects, and different opinions from people could measure this diversity to some extent.

NDCG@10 automatically incorporates the degree of relevance when evaluating methods. The ground-truth score of NDCG@10 is accumulated from all annotators, with *not similar* corresponding to 0 points, *somehow similar* to 1 point and *similar* to 2 points, resulting in a $[0 - 6]$ grading scale.

Except for Prec@10-relax, the measures are in general accuracy oriented, without much consideration of diversity. We give more analysis on diversity in Section 6.1 and 6.2, where we do content analysis, and use an entropy-based measure to study diversity.

[5]https://www.mturk.com/mturk

5.5 Method Evaluation

Baselines. Existing methods on entity search rely on labeled data set, such that methods like matrix factorization [25] and learning to rank [9] could be used. In absence of labeled data, we compare our proposed methods to retrieval-based ones.

It can be observed that individual rankers, e.g., TITLE, are an instantiation of a KL-divergence similarity, a simple but effective retrieval method [41]. Hence, the individual rankers could naturally serve as baseline methods. However, we only include COMBINER and EXPANDER for human judgments. First, many of the individual rankers are sparse, and exhibit performance inferior to COMBINER and EXPANDER in preliminary studies. Second, as an an ensemble of individual rankers, COMBINER was shown to be a highly effective baseline in prior work [8]. Third, we still evaluate the effectiveness of individual rankers by doing feature analysis in Section 6.4.

COMBINER++. We combine the ranking list of COMBINER and EXPANDER still by Reciprocal Rank Fusion [8], but with a weighted version of Equation 5: $sc(S,T) = \sum_j \frac{w_j}{r^j_{S,T}+K}$, where w_j is the weight of $ranker_j$, optimized using fold splits defined above. Note that rankers here only refer to COMBINER and EXPANDER, excluding individual rankers used in Equation 5.

Learning to rank. When human judgments are finally available after we evaluate competing methods, we include supervised learning methods for further experiments: we utilize learning-to-rank techniques to analyze and compare the importance of individual rankers. The evaluation for these methods is very similar to 10-fold cross validation, with the exception that only 1 fold, rather than 9, is used for training each time. This is to see whether the performance of our methods could be enhanced with minimal supervision, and which feature categories contribute most to such enhancement. For each instance in the dataset, a feature value corresponds to a score given by one of the rankers, including COMBINER, EXPANDER, and all the individual rankers. We experiment with a set of learning-to-rank algorithms that are provided by the library RankLib [10]. For all the methods, we use the default parameters without any additional tuning.

6. EXPERIMENT RESULTS

Based on the annotations from AMT, the performance of proposed methods are calculated and displayed in Table 3. Since all learning-to-rank methods outperform the rest, we only show the top three of them, listed in the last three rows of Table 3.

Table 3: Evaluation results averaged across four regions.

	NDCG	Prec@10-vote	Prec@10-relax
COMBINER	58.86	41.95	75.20
EXPANDER	52.66	32.80	78.35*
COMBINER++	60.08*	42.31*	77.01*
RankBoost	62.85*	44.57**	82.53**
Linear regression	63.50*	44.45**	82.98**
Coordinate Ascent	**63.71****	**44.71****	**82.99****

*(**) indicates the improvement over COMBINER is statistically significant according to paired t-test at the significance level of 0.05(0.01). The last three rows are learning-to-rank methods.

Comparing methods COMBINER and EXPANDER, we obtain mixed results. Judging by NDCG and Prec@10-vote, which care more about agreement among annotators, COMBINER outperforms EXPANDER. On the other hand, EXPANDER is better on Prec@10-relax, which reflects more subjectivity of the annotators. Given these results, we suspect that COMBINER acts more conservatively by giving safe and accurate results. In contrast, EXPANDER explores more, suggesting diverse and semantically relevant events. To verify our hypothesis, we analyze the content of the evaluation results, and find a measure to quantify diversity. The analysis is detailed in the following subsections.

When applying rank fusion to COMBINER and EXPANDER, COMBINER++ achieves a balance between accuracy and diversity. It surpasses COMBINER significantly on all three metrics, and is comparable to EXPANDER on Prec@10-relax. This means COMBINER++ provides a way to tradeoff between relevance and diversity, and users in real scenarios could adjust the trade-off parameter for their personal use.

When we are able to learn the importance of different rankers by learning to rank, there is a significant improvement over methods without supervision. The exact contribution of these rankers, or feature categories, will be discussed in Section 6.4. With regard to the comparison among learning-to-rank algorithms, simple models (e.g., Coordinate Ascent and Linear regression) work better than complex ones (e.g., LambdaMART and ListNet), possibly due to: (1) the low dimensionality of feature space, which equals to the number of rankers; (2) the small size of training set, as only 1 fold of the data is used for training each time.

6.1 Content analysis

In order to further understand the behaviors of COMBINER and EXPANDER, we sampled some annotated results, and look through the exact events being retrieved.

Table 4: Example events where COMBINER excels.

Query event: Sacramento California Capitol Apocalypse 5k Zombie Run 2013			
COMBINER		EXPANDER	
Score	Events	Score	Events
6	8K Golden Gate Double Adventure Run (Running)	6	Blacklight Run - San Jose (Running)
5	Run on the Parkway (Running)	2	Almaden Hacienda Hills (Hiking)
5	Lil Mud Runner Kids/ Family Mud Run (Running)	2	Fitch Mountain Footrace (Footracing)
Query event: Let's Go See "Only Lovers Left Alive" | 7:15 Pm Show			
COMBINER		EXPANDER	
Score	Events	Score	Events
4	Birdman (Movie)	3	Jersey Boys (Drama)
2	Rocky Horror Show: The Cult Classic Live (Show)	2	Happy Christmas Movie, Q&A With Joe Swanberg (Movie)
2	Happy Christmas Movie, Q&A With Joe Swanberg (Movie)	0	Foo Fighters (Music)

We first analyze cases where COMBINER outperforms EXPANDER in terms of NDCG. This happens when query events are more frequent and common, as Table 4 shows. The score is the ground-truth score used to compute NDCG, which is accumulated from 3 annotators, where *similar* and *somewhat similar* correspond to 2 and 1 points, respectively. For readability, event types are appended to the end of each event, wrapped in parenthesis (which are hidden from annotators).

The first query event in Table 4 is about running. Top events ranked by COMBINER all contain the keyword run, which are accurate but not diverse enough. Though EXPANDER ranks a running event in the first place too, it lists other sports-related events like hiking and mountain footracing, which are however labeled as not similar by the majority of annotators. This results from anchoring bias, noted in several studies on human annotations [36, 34]. When more relevant events are present – events with "run" in titles, the relevance of hiking and footracing events could be significantly reduced in annotator's view.

Another example comes from a movie event. In contrast to COM-

BINER that finds many movie events, EXPANDER in addition ranks events about drama and a rock band on top. Though both movies and rock bands belong to the category of entertainment, none of the annotators think they are relevant, which is, again, likely caused by the anchoring bias.

Table 5: Example events where EXPANDER excels.

Query event: Hiking Trip: Armstrong Woods With Marcia 7-8 Miles			
COMBINER		EXPANDER	
Score	Events	Score	Events
1	Health 2.0 Sacramento Presents: Telemedicine 2015, A Multi Billion$ Market (Healthcare)	4	Rock Climbing: Yosemite: Snake Dike (Climbing)
1	Get The Led Out - Tribute Band (Music)	5	Sunol Wilderness Backpacking Overnight! (Backpacking)
1	Prevention, & Treatment Of Poison Oak (Plants)	6	Caving: Church Cave, Kings Canyon National Park (Caving)
Query event: Semicon 2015			
COMBINER		EXPANDER	
Score	Events	Score	Events
1	Cricket World Cup- Semi Finals (Cricket)	1	Frontiers In Optics: The 99Th Osa Annual Meeting And Exhibit/Laser Science (Optics)
1	Benefit Show For Raul And His Girls (Show)	4	Optical Fiber Communication Conference And Expo (Optics)
1	Career Night at College Park High School (Career)	0	Sf: Word Of Mouth - Founder Community (Entrepreneur)

When events are less frequent, EXPANDER begins to demonstrate its strength in semantics, as Table 5 exemplifies. In the first query of a hiking event, COMBINER fails to find any keyword synonymous to hiking, and is almost off-topic. Contrarily, EXPANDER suggests rock climbing, backpacking overnight, and caving, which are all events that could be taken in mountains. Looking at the scores, some annotators label hiking as weakly connected to healthcare, music, and plants. This behavior is also attributed to anchoring bias [36, 34] – when there are less relevant events in the candidate list, annotators tend to give more credits to events, even though they are not that relevant.

Another example is about a semiconductor expo, where COMBINER is off-topic again and EXPANDER finds events on laser science and optical fiber. Unfortunately, many annotators fail to recognize their connections. Note that it is possible that annotators know this connection in the second event, while not in the first. This happens when the two events fall into different HITs, which are completed by different annotators.

In this analysis, we find that the task of annotating event similarities is both difficult and subjective. Due to the anchoring bias, annotators tend to raise their judgment threshold of relevance when relevant events abound, assigning a lower score to events that are somewhat similar. While in the opposite scenario, where fewer relevant events are available, annotators are inclined to reduce their threshold. This cannot be avoided if multiple candidates are presented to the annotator at the same time. However, if we change the evaluation process by showing query-candidate pairs one at a time, other bias could be introduced. For example, pairs labeled by different annotators might not be comparable, due to subjectivity. Even for the same annotator, they might change their mind as more pairs are revealed.

Despite the difficulty of annotation, some insights can still be gained from the judged results. COMBINER is good at finding out closely related events for common event themes, while EXPANDER is able to retrieve semantically relevant candidates for rarer events. Generally speaking, common query events are more frequent in our randomly sampled evaluation set. As a result, COMBINER, overall, performs better on the metrics that care more about accuracy, i.e., NDCG and Prec@10-vote. However, integration of COMBINER and EXPANDER via COMBINER++ and learning to rank methods (see Table 3) can further improve performance.

6.2 Diversity analysis

Content analysis shows that EXPANDER is more exploratory, giving more diverse results. We want to examine whether this observation can be generalized to the entire dataset. Since it is neither realistic nor objective to manually look through all the results, it is necessary to find a way to quantify diversity. To this end, we utilize entropy, a widely accepted measure to quantify the amount of information contained in a message. The intuition is that, if the candidate list of a query contains more information, the events in the list are more diverse. In particular, we collect event titles for the retrieved candidates of each query event. Given the set of title text, word distribution can be computed, based on which we calculate entropy: $H = -\sum_{i=1} p(w_i) \cdot \log p(w_i)$, where w_i is the i-th word, and the probability $p(w_i)$ is estimated by its occurrence in the title set. The entropy is then averaged over all query events, as shown in Table 6.

Table 6: Average entropy of candidate events.

Method	Avg entropy	Method	Avg entropy
QUERY	1.36	SURROUND	6.89
ANCHOR	5.21	COMBINER	7.46
TAXONOMY	6.14	COMBINER++	7.61
TITLE	6.64	EXPANDER	8.00

As we expected, EXPANDER achieves the highest entropy, corroborating our thoughts on its diversity. COMBINER scores much lower, showing that it is indeed giving results that are alike to each other. COMBINER++ attempts to balance accuracy and diversity, whose successfulness is reflected by its entropy score.

If we use TITLE as the reference method, COMBINER gains higher entropy, meaning that combining a set of individual rankers could improve the diversity. TAXONOMY and SURROUND obtain scores very close to TITLE, indicating that their behaviors are relatively normal. The entropy for QUERY is extremely low, largely due to the lack of queries for many events, which will be clearer when we analyze recall in Section 6.3. This recall problem also applies to ANCHOR, but is less severe.

6.3 Recall analysis

Evaluating recall requires manual judgment of a large number of events, which is empirically impossible. Hence we approximate it by computing, for each query event, the number of candidate events retrieved by each method, with the maximum set to 100.

Table 7: Averaged number of candidate events.

Method	Avg candidates	Method	Avg candidates
QUERY	17.92	TITLE	89.75
ANCHOR	63.99	COMBINER	99.79
TAXONOMY	72.63	EXPANDER	100
SURROUND	83.37	COMBINER++	100

As Table 7 shows, EXPANDER and COMBINER++ are always able to retrieve the maximum number of candidates. By combining signals from various feature categories, COMBINER is close to perfection. It is somewhat surprising that TITLE could fetch fairly large number of candidates. Analysis shows that this is caused by matches of unimportant but frequent words, which are prevalent as we are not employing a very aggressive stopword removal. For example, TITLE-ranker could retrieve *soccer night* for *movie night*, since the matching of word *night*. As we discussed in diversity analysis, QUERY has the lowest recall, followed by ANCHOR.

6.4 Feature analysis

In order to analyze the importance of individual feature category, we perform an ablation study. We rerun the Coordinate Ascent, in our case the best performing learning-to-rank algorithm, by removing one feature category at one time, while keeping all other features present. We do not analyze TITLE-ranker here since events with title matches are removed to encourage annotators to consider semantic relevance. Table 8 shows the percentage of performance reduction relative to the method using all feature categories. In addition to removing features, we also consider using a single category alone, with all other features excluded. The results are displayed in Table 9.

Table 8: Performance reduced by removing a feature category.

	NDCG	Prec@10-vote	Prec@10-relax
QUERY	0.37%	0.77%	0.11%
ANCHOR	0.42%	0.31%	0.42%
SURROUND	2.47%	3.75%	0.45%
TAXONOMY	9.47%	10.58%	1.36%

Table 9: Performance reduced by using one feature category.

	NDCG	Prec@10-vote	Prec@10-relax
QUERY	11.44%	13.38%	4.45%
ANCHOR	11.13%	11.97%	3.24%
SURROUND	8.74%	9.81%	2.76%
TAXONOMY	1.97%	3.27%	2.54%

Table 8 and 9 generally give consistent results. First, all numbers being positive indicates that individual rankers are outperformed by algorithms combining them. By integrating information from different feature categories, our methods indeed beat baselines that use a single feature source.

Focusing on specific feature categories, TAXONOMY plays a vital role in achieving good performance, which means that the taxonomic categories provide an important source for COMBINER to retrieve semantically relevant events. Features extracted from anchor and surrounding text could be quite noisy, leading to their lower performance. In addition to noisy features, ANCHOR could also suffer from low recall.

QUERY is least important, mainly because of the recall problem. In this learning algorithm, each feature category works independently. Contrarily in the case of EXPANDER, identical features from different categories are collapsed into one feature, and messages can propagate through graph. Under such circumstance, the problem of sparseness could be alleviated and features extracted from query could potentially make a higher contribution.

It is also interesting to compare the performance drop between metrics. Unlike NDCG and Prec@10-vote, the decrease of Prec@10-relax is much smaller. This is due to the existence of larger number of relevant instances in the relaxed version of Prec@10. As a result, it is easier for methods to attain a relatively high Prec@10-relax.

6.5 Computational cost

EXPANDER can empirically finish training all the web events in less than 30 minutes, as it keeps constant number of labels per node. When new events come, EXPANDER does not need to be trained over the entire graph, as incremental inference could be performed [32].

In this paper we purposefully focused on the effectiveness of the proposed event retrieval methods, and not on their efficiency. For online retrieval, COMBINER can be straight-forwardly implemented by indexing event features in an inverted index. EXPANDER can be updated periodically by incremental inference [32], while caching labels for existing events and features. At new event arrival, we may infer its label by looking up the cached labels of its features in the graph.

We leave further exploration of event indexing and retrieval strategies as an interesting avenue for future work.

7. CONCLUSION AND FUTURE WORK

In this paper, we consider the problem of related event discovery. In particular, we study events with Schema.org markup. Due to the temporal and local nature of events, ground-truth information is hard to acquire. In addition, limited information can be extracted from the event itself, while its contextual information is sparse and noisy. In face of these challenges, we propose a graph-based framework to retrieve a ranked list of events for a source event. Our framework is unsupervised, and addresses the semantic relevance in a self-contained manner.

We evaluate our framework using Amazon Mechanical Turk, and demonstrate the feasibility of its two proposed variants: COMBINER and EXPANDER. Analysis shows that COMBINER is more precise, while EXPANDER provides more diverse results. Their combination balances the two aspects, and the effectiveness can be further improved when we are able to learn the importance of different feature categories with minimum supervision. Feature analysis shows that our framework significantly outperforms traditional retrieval methods based on individual feature categories, which often suffer from recall problem.

8. REFERENCES

[1] S. Auer and J. Lehmann. What have innsbruck and leipzig in common? extracting semantics from wiki content. In *Proc. of ESWC*, pages 503–517. 2007.

[2] K. Balog, P. Serdyukov, and A. P. d. Vries. Overview of the trec 2010 entity track. Technical report, DTIC Document, 2010.

[3] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for Youtube: Taking random walks through the view graph. In *Proc. of WWW*, pages 895–904, 2008.

[4] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. 2006.

[5] C. Bizer, S. Auer, G. Kobilarov, J. Lehmann, and R. Cyganiak. Dbpedia–querying wikipedia like a database. In *Proc. of WWW*, pages 8–12, 2007.

[6] R. Blanco, P. Mika, and S. Vigna. Effective and efficient entity search in rdf data. In *ISWC*, pages 83–97. 2011.

[7] J. G. Conrad and M. H. Utt. A system for discovering relationships by feature extraction from text databases. In *SIGIR'94*, pages 260–270, 1994.

[8] G. V. Cormack, C. L. Clarke, and S. Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proc. of SIGIR*, pages 758–759, 2009.

[9] L. Dali, B. Fortuna, T. T. Duc, and D. Mladenić. Query-independent learning to rank for rdf entity search. In *The Semantic Web: Research and Applications*, pages 484–498. 2012.

[10] V. Dang. The Lemur project – RankLib. *Available: http://sourceforge.net/p/lemur/wiki/RankLib*.

[11] A. Q. de Macedo and L. B. Marinho. Event recommendation in event-based social networks. In *Hypertext, Social Personalization Workshop*, 2014.

[12] A. P. De Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the inex 2007 entity ranking track. In *Focused Access to XML Documents*, pages 245–251. 2008.

[13] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker. Hierarchical link analysis for ranking web data. In *The Semantic Web: Research and Applications*, pages 225–239. 2010.

[14] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and ranking knowledge on the semantic web. In *ISWC*, pages 156–170. 2005.

[15] S. Dooms, T. De Pessemier, and L. Martens. A user-centric evaluation of recommender algorithms for an event recommendation system. In *Proc. RecSys: Workshop on Human Decision Making in Recommender Systems and UCERSTI*, pages 67–73, 2011.

[16] J. Foley, M. Bendersky, and V. Josifovski. Learning to extract local events from the web. In *Proc. of SIGIR*, pages 423–432, 2015.

[17] D. Graus, M. Tsagkias, W. Weerkamp, E. Meij, and M. de Rijke. Dynamic collective entity representations for entity ranking. *WSDM*, page 16, 2016.

[18] R. V. Guha, D. Brickley, and S. MacBeth. Schema.org: Evolution of structured data on the web. *Queue*, 13(9):10:10–10:37, Nov. 2015.

[19] A. Harth, S. Kinsella, and S. Decker. *Using naming authority to rank data and ontologies for web search*. Springer, 2009.

[20] A. Hogan, A. Harth, and S. Decker. Reconrank: A scalable ranking method for semantic web data with context. 2006.

[21] C. Kang, D. Yin, R. Zhang, N. Torzec, J. He, Y. Chang, and Y. Labs. Learning to rank related entities in web search. *Neurocomputing*, 2015.

[22] M. Kayaalp, T. Ozyer, and S. Ozyer. A collaborative and content based event recommendation system integrated with data collection scrapers and services at a social networking site. In *ASONAM'09*, pages 113–118, 2009.

[23] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In *Proc. of RecSys*, pages 185–192, 2013.

[24] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. Event-based social networks: linking the online and offline social worlds. In *Proc. of SIGKDD*, pages 1032–1040, 2012.

[25] A. Q. Macedo, L. B. Marinho, and R. L. Santos. Context-aware event recommendation in event-based social networks. In *Proc. of RecSys*, pages 123–130, 2015.

[26] E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola. Collaborative future event recommendation. In *Proc. of CIKM*, pages 819–828, 2010.

[27] D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In *Proc. of CIKM*, pages 731–740, 2007.

[28] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object ranking in the web of data. *Proc. WWW*, pages 771–780, 2010.

[29] Z. Qiao, P. Zhang, C. Zhou, Y. Cao, L. Guo, and Y. Zhang. Event recommendation in event-based social networks. In *Proc. of AAAI*, 2014.

[30] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *Proc. of WSDM*, 2013.

[31] H. Raghavan, J. Allan, and A. McCallum. An exploration of entity models, collective classification and relation description. 2004.

[32] S. Ravi and Q. Diao. Large scale distributed semi-supervised learning using streaming approximation. In *Proc. of AISTATS*, 2016.

[33] J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[34] F. Scholer, D. Kelly, W.-C. Wu, H. S. Lee, and W. Webber. The effect of threshold priming and need for cognition on relevance calibration and assessment. In *Proc. of SIGIR*, pages 623–632. ACM, 2013.

[35] P. Serdyukov, H. Rode, and D. Hiemstra. Modeling multi-step relevance propagation for expert finding. In *Proc. of CIKM*, 2008.

[36] M. Shokouhi, R. White, and E. Yilmaz. Anchoring and adjustment in relevance estimation. In *Proc. of SIGIR*, pages 963–966. ACM, 2015.

[37] P. P. Talukdar and K. Crammer. New regularized algorithms for transductive learning. In *Proc. of ECML PKDD*, pages 442–457, 2009.

[38] R. van Zwol, L. G. Pueyo, M. Muralidharan, and B. Sigurbjornsson. Ranking entity facets based on user click feedback. In *Proc. of ICSC*, pages 192–199, 2010.

[39] Y. Wang, R. Ji, and S.-F. Chang. Label propagation from imagenet to 3d point clouds. In *Proc. of CVPR*, pages 3135–3142, 2013.

[40] W. Weerkamp, K. Balog, and E. Meij. A generative language modeling approach for ranking entities. In *Advances in Focused Retrieval*, pages 292–299. 2009.

[41] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proc. of CIKM*, pages 403–410, 2001.

[42] J. Zhu, D. Song, and S. Rüger. Integrating document features for entity ranking. In *Focused Access to XML Documents*, pages 336–347. 2008.

[43] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.