Ranking Archived Documents for Structured Queries on Semantic Layers

Vaibhav Kasturia, Pavlos Fafalios, and Wolfgang Nejdl

L3S Research Center, University of Hannover, Germany
{kasturia, fafalios, nejdl}@l3s.de

Abstract. Archived collections of documents (like newspaper and web archives) serve as important information sources in a variety disciplines, including Digital Humanities, Historical Science, and Journalism. However, the absence of efficient and meaningful exploration methods still remains a major hurdle in the way of turning them into usable sources of information. Semantic Layers allow describing and publishing metadata and semantic information about the archived documents in a standard format (RDF), which in turn can be queried through a semantic query language (SPARQL). This allows running advanced queries by combining metadata of the documents (like publication date) and content-based semantic information (like entities mentioned in the documents). However, the results returned by structured queries can be numerous and also they all equally match the query. There is thus the need to rank the returned results in order to identify and promote the most important ones. In this paper, we formalize the problem of *ranking archived documents for structured queries*, we distinguish two main types of queries for this problem, and we propose a baseline ranking model that jointly considers the following aspects: i) the relativeness of documents to entities, ii) the timeliness of documents, and iii) the relations among the entities. The results of an extensive experimental evaluation on a news archive are promising.

1 Introduction

Despite the increasing number of digital archives worldwide (like newspaper and web archives), the absence of efficient and meaningful exploration methods still remains the major bottleneck in the way of turning them into usable information sources [?]. Semantic models try to solve this problem by offering the means to describe metadata and semantic information about a collection of archived documents in the standard RDF format. A repository of such data, called Semantic Layer [?], allows running advanced queries which combine metadata of the documents (like publication date) and content-based semantic information (like entities mentioned in the documents). As an example, we can access a Semantic Layer over a newspaper archive and find articles of a specific time period discussing about a specific category of entities (e.g., *philanthropists*) or about entities sharing some characteristics (e.g., *lawyers born in Germany*).

Such advanced information needs can be directly expressed through structured (SPARQL) queries or through user-friendly interactive interfaces which transparently transform user interactions to SPARQL queries (e.g., a faceted browsing interface [**?**]).

However, the results returned by such queries can be numerous and moreover they all equally match the query. Thus, there arises the need for an effective method to rank the returned results for discovering and promoting the most important ones. As an example, when requesting articles from a news archive mentioning a specific entity, some of the returned articles may be about that entity while other articles may just mention the entity without it being their main topic. Thus, an effective ranking method should consider the different factors that affect the importance of documents to the information need, relying at the same time only on the data available in the semantic layer (without accessing documents' full contents).

Although there is a plethora of works on both ranking archived documents for keyword-based queries and ranking structured data (subjects and objects) in knowledge graphs, the problem of ranking archived documents for structured queries in knowledge graphs has not yet been recognized and studied. In this paper, we address this gap by first introducing and formalizing this task and then proposing a ranking model for the problem at hand. The proposed model jointly considers the following aspects: i) the *relativeness* of a document with respect to the entities of interest, ii) the *timeliness* of document's publication date, iii) the temporal *relatedness* of the entities of interest with other entities mentioned in the document. The idea is to promote documents that mention the entities of interest many times, that have been published in important (for the entities of interest) time periods, and that mention many other entities co-occurring frequently with the entities of interest in important time periods. For example, in case we want to rank articles of 1990 discussing about *Nelson Mandela*, we want to favor articles that i) mention *Nelson Mandela* multiple times in their text, ii) have been published in important time periods for *Nelson Mandela* (e.g., February 1990 since during that period he was released from prison), and iii) mention other entities that seem to be important for *Nelson Mandela* during important time periods (e.g., *Frederik Willem de Klerk* who was South Africa's State President in February 1990).

In a nutshell, in this paper we make the following contributions:

– We formulate and formalize the problem of ranking archived documents for structured queries over semantic layers.
– We proposed a ranking model for the problem at hand that ... ...
– Due to lack of evaluation datasets for our problem, we have created a new ground truth dataset for a news archive which we make publicly available.
– The evaluation results show that the proposed ranking model ...

The rest of the paper is organized as follows: Section 2 motivates our work and presents related literature. Section 3 presents basic algorithms for ranking documents, entities, and time periods. Section 4 tries to describe a probabilistic approach for the problem at hand. Section **??** evaluates the algorithms and their combinations. Finally, Section ?? concludes the paper and presents interesting directions for future research.

2 Background and Related Works (DRAFT)

In this section, we first provide a background of the *Open Web Archive* Data Model in which we describe the data model along with a sample depiction of a non-versioned article as in the semantic layer. Then we mention the different works related to ranking of archived documents, ranking in knowledge graphs and ranking documents in knowledge graphs.

2.1 The "Open Web Archive" Data Model

In our previous work [?] we introduced an RDF/S data model to describe the semantic information and metadata about the documents of a web archive. This model, which we call as *Open Web Archive* data model[1], is depicted in Figure 1. We re-use elements from many other existing data models and define 2 new classes and 3 new properties. An archived document is represented using the class `owa:ArchivedDocument`. Further, the archived document may or may not be linked with some versions (i.e., instances of `owa:VersionedDocument`). Versions pages for billions of web sites can be found on the Internet Archive. In contrast, the New York Times corpus [?] that we try applying our ranking models on does not contain versions.

We associate an archived document with three main elements: i) metadata information, like format(mime type), date of capture/publication, and document title, ii) links to other documents (web pages), archived or not, and iii) set of annotations. Terms from the Dublin Core Metadata Initiative[2] were used to describe some of the metadata. Annotations were described by exploiting the Open Annotation Data Model [?] and the Open Named Entity Extraction (NEE) Model [?].

The Open Annotation Data Model contains an RDF-based framework specification for creating associations (annotations) between related resources, while the Open NEE Model is an extension that allows describing the result of an entity extraction process. An annotation has a *target*, which is an archived document in our case, and a *body* which is a concept, entity or event mentioned in the document. An archived document can be directly related with an entity, concept or event by exploiting the property "`mentions`" of schema.org[3] for reducing the number of derived triples. A concept, entity or event can be associated with information like its name, a confidence score, its position in the document, and a resource (URI). The URI enables to retrieve additional information from the Linked Open Data (LOD) cloud [?] (like properties, relations with other entities, etc.).

[1] Specification publicly available at: `http://l3s.de/owa/`

[2] `http://dublincore.org/`

[3] `http://schema.org/mentions`

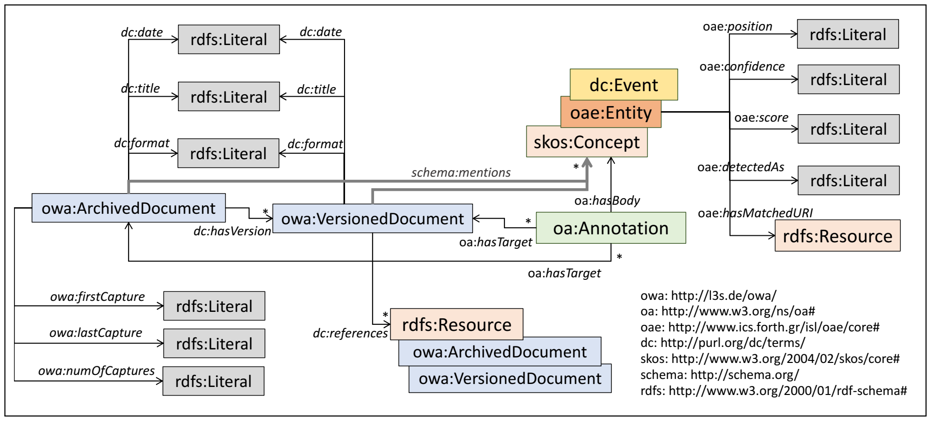

Fig. 1: The *Open Web Archive* data model.

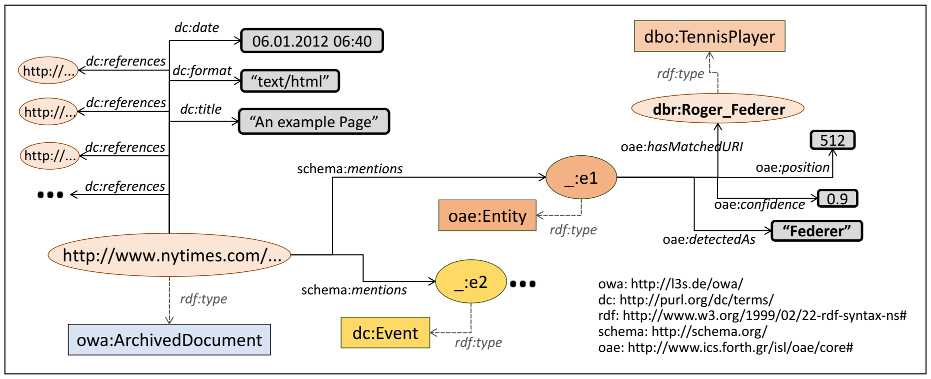

Fig. 2: Describing an archived article (non-versioned) using the *Open Web Archive* data model.

Figure 2 depicts an example of an archived non-versioned article. Some of its metadata values (date, format, title), its references to other web pages, and its annotations are visible. We notice that the entity name "Federer" was identified in that document. It is also visible that the entity has been linked with the DBpedia resource corresponding to the tennis player *Roger Federer*. By accessing DBpedia, we can now retrieve more information about this entity like its birth date, an image, a description in a specific language, etc. Description of versioned pages using *Open Web Archive* data model is not mentioned in this paper since our work is mainly focussed on ranking on non-versioned pages. Details of the construction process of Semantic Layers have also not been included. Interested readers should refer to our previous work [?] for a more detailed description of this model.

2.2 Related Work (DRAFT)

Ranking of Archived Documents We mention some of the recent works in the field of ranking of Archived Documents using full content based search or extraction of metadata.

Tempas [?] is a keyword-based search system that exploits a social bookmarking service for temporally searching a web archive by indexing tags and time. It allows temporal selections for search terms, ranks documents based on their popularity and also provides query recommendations.

Singh et al. [?] introduce the notion of *Historical Query Intents* and model it as a search result diversification task which intends to present the most relevant results (for free text queries) from a topic-temporal space. For retrieving and ranking historical documents (e.g., news articles), the authors propose a novel retrieval algorithm, called HistDiv, which jointly considers the aspect and time dimensions.

Expedition [?] is a time-aware search system for scholars. It allows users to search articles in a news collection by entering free-text queries and choosing from four retrieval models: Temporal Relevance, Temporal Diversity, Topical Diversity, and Historical Diversity.
Difference of our approach. Our approach provides ranking using structured queries rather than keyword-based queries. To provide ranking, we do not access the content of the articles rather a layer composed of RDF triples which contains metadata, links and entities extracted from the articles.

Ranking in Knowledge Graphs Most of the Ranking Approaches on Knowledge Graphs are adaptations of the existing approaches for unstructured data.

Dali et. al. [?] propose a query independent Learning to Rank(LTR) approach for RDF entity search. They use RDF graph extracted features, search engine based features and centrality-based features and compare them to target features. Latifi and Nematbaksh [?] also use the same approach but suggest the use of Information Content(IC) feature to reduce ranking time of the system.

OntologyRank algorithm by Ding et al. introduced in [?] and mentioned further in [?] finds use in the Semantic web search engine *Swoogle*. It identifies Semantic Web Ontologies(SWOs) in Semantic Web Documents(SWDs) and further ranks terms in an Ontology based on their popularity. AKTiveRank [?] by Alani et. al. is another ontology ranking approach which relies on their importance to a given query and uses the semantic web search engine *Swoogle* [?] to get a list of ontologies that need to be ranked. SemRank [?] designed by Anwanyu et. al. ranks relationships in SWDs.

Regarding PageRank based approaches, PopRank [?] assigns weights to links among Web Objects depending on the relationship types between objects. This system extracts a subset of the graph based on domain and then assigns link weights to the subgraph using expert generated Ranking Lists. Harth et. al. [?] perform ranking by assigning weights to all links in the graphs based on authority or provenance of data source and calculating PageRank for the whole graph even before the query is entered. This approach is in complete contrast to ReConRank [?], an algorithm that performs dynamic ranking only analysing the result data that matches the user query and which considers the ratio of all the links received as in-links in order to reduce the number of iterations. RareRank [?] is another algorithm where the random component in the PageRank model is replaced by a more deterministic component based on the domain of search in order to reduce randomness in a graph. DBPediaRanker [?] first dereferences and explores all nodes in the DBPedia graph belonging to the same domain given a query. Then by checking whether a strong relation exists between two resources, it creates a contextualized weighted graph. YAGO-NAGA introduced in Kasneci et. al. [?] and extended to include keyword based search in Elbassouni et. al. [?] is a semantic search system based on Language Model that performs ranking based on the notions of Informativeness, Compactness and Confidence. DING [?] calculates rank in three steps: first the global dataset rank, then the entity rank and finally the global ranking which is a combination of the of both the global dataset rank and the entity rank. Fafalios and Tzitzikas [?] integrate classical Web with the LOD Web using PageRank like algorithm to provide users with semantic context to help them save time in exploratory search scenarios.

Considering link-based approaches other than PageRank adaptations, NOC-ORDER [?] introduced by Graves et. al. is an adaptation of *All-Pairs Shortest Path* algorithm that ranks nodes in an RDF graph based on centrality feature.

Difference of our approach. Our work ranks documents rather than entities, relationships [?] or ontologies ([?], [?] and [?]). Unlike [?] and [?], our approach does not require any training data. It does not use any kind of link analysis like in [?], [?], [?], [?], [?], [?] and [?]. Further, its not domain specific like [?] and [?], doesn't rely on provenance of data sources as in [?], [?] and [?] and produces the ranking at query-execution time unlike [?].

Ranking Documents in Knowledge Graphs Little work has been done on ranking documents in knowledge graphs.

Swoogle, the Semantic web search engine, uses the OntologyRank algorithm [?], [?] as mentioned in the previous sub-section and operates at the document level as well as the term and RDF graph level. It calculates relevance of Semantic Web Documents based on whether a document imports, uses, extends definition of the other document's terms or makes assertions about the individuals defined by the other document.

Difference of our approach. Our approach does not judge relevance of documents based on the relationships between the documents, because our problem is to rank a set of relevant documents rather than rank on the relevance of documents. To the best of our knowledge, we are the very first authors to perform such ranking using Semantic Models/Layers.

3 Ranking Documents for Structured Queries

In this section, we formalize the problem of ranking documents returned by structured (SPARQL) queries and define a probabilistic ranking model. First we introduce the required notions and notations.

3.1 Notions and Notations

Entities. In our problem, an *entity* is anything with a separate and meaningful existence that also has an identity expressed through a reference in a knowledge base (e.g., a Wikipedia/DBpedia URI). This does not only include persons, locations, organizations, etc., but also events (e.g., *US 2016 presidential election*) and more abstract concepts such as *democracy* or *apartheid*. Let E be a finite set of entities, e.g., all Wikipedia entities, where each entity $e \in E$ is associated with a unique URI in the reference knowledge base.

Documents and extracted entities. Let D be a set of documents (e.g., a set of news articles) published within a set of time periods T_D of fixed granularity Δ (e.g., day). For a document $d \in D$, let $t_d \in T_D$ be the time period of granularity Δ in which d was published, while for a time period $t \in T_D$, let $docs(t) \subseteq D$ be the set of all documents published within t, i.e., $docs(t) = \{d \in D \mid t_d = t\}$.

Let now $ents(d) \subseteq E$ be all entities mentioned in d. Inversely, for an entity $e \in E$, let $docs(e) \subseteq D$ be all documents that mention e, i.e., $docs(e) = \{d \in D \mid e \in ents(d)\}$.

3.2 Problem Definition

Given a corpus of documents D, a set of entities E_D mentioned in documents of D, and a SPARQL query Q requesting documents from D published within a *time period* $T_Q \subseteq T_D$ and related to one or more query entities $E_Q \subseteq E_D$ with logical `AND` (mentioning all the query entities) or `OR` (mentioning at least one of the query entities) semantics, the problem is how to rank the documents $D_Q \subseteq D$ that (equally) match Q.

Figure 3 shows an example SPARQL query requesting documents published in 1990 discussing about the entities *Nelson Mandela* and *Frederik Willem de Klerk* (`AND` semantics), while the query in Figure 4 requests articles of 1990 discussing about *state presidents of South Africa* (`OR` semantics). Our objective is to rank the results returned by such SPARQL queries.

3.3 Probabilistic Modeling

We model and estimate the probability to select a document $d \in D_Q$ given the query entities E_Q, the time period of interest T_Q, and other entities E_{D_Q} mentioned in the retrieved documents. Specifically, we model the following aspects:

– the *relativeness* of a document with respect to the query entities

```
SELECT DISTINCT ?article WHERE {
  ?article dc:date ?date FILTER(?date >= "1990-01-01"^^xsd:date &&
                                ?date <= "1990-12-31"^^xsd:date) .
  ?article oae:mentions ?entity1, ?entity2 .
  ?entity1 oae:hasMatchedURI  <http://dbpedia.org/resource/Nelson_Mandela> .
  ?entity2 oae:hasMatchedURI  <http://dbpedia.org/resource/F._W._de_Klerk> }
```

Fig. 3: SPARQL query for retrieving articles of 1990 discussing about *Nelson Mandela* and *Frederik Willem de Klerk* (`AND` semantics).

```
SELECT DISTINCT ?article WHERE {
  SERVICE <http://dbpedia.org/sparql> {
    ?p dc:subject <http://dbpedia.org/resource/Category:State_Presidents_of_South_Africa> }
  ?article dc:date ?date FILTER(?date >= "1990-01-01"^^xsd:date &&
                                ?date <= "1990-12-31"^^xsd:date)
  ?article oae:mentions ?entity .
  ?entity oae:hasMatchedURI  ?p }
```

Fig. 4: SPARQL query for retrieving articles of 1990 discussing about *state presidents of South Africa* (`OR` semantics).

- the *timeliness* of a document with respect to its publication date
- the *relatedness* of a document with respect to its reference to other entities related to the query entities

The idea is to promote documents that mention the query entities many times in their contents, have been published in important (for the query entities) time periods, and mention many other entities that co-occur frequently with the query entities in important time periods. For example, in case we want to rank articles of 1990 discussing about *Nelson Mandela*, we want to favor articles that i) mention *Nelson Mandela* multiple times in their text, ii) have been published in important (for *Nelson Mandela*) time periods (e.g., February of 1990 since during that period he was released from prison), and iii) mention other entities that seem to be important for Nelson Mandela during important time periods (e.g., *Frederik Willem de Klerk* who was South Africa's State President in 1990).

Relativeness

We consider that if the query entities are mentioned multiple times within a document, the document should receive a high score (since the document's topic may be about these entities). The term frequency (in our case entity frequency) is a classic numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [?].

We first define a *relativeness* score of a document $d \in D_Q$ based on the *frequency* of the query entities in d. First, let $count(e, d)$ be the number of occurrences of e in document d. For the case of `AND` semantics ("$\wedge$"), the score is defined as follows:

$$score^f_\wedge(d, E_Q) = \frac{\sum_{e \in E_Q} count(e, d)}{\sum_{e' \in ents(d)} count(e', d)} \tag{1}$$

Notice that the score of a document will be 1 if it contains the query entities and no other entity. For the case of `OR` semantics ("$\vee$"), we can also consider the number of query entities mentioned in the document (since a document does not probably contain all the query entities as in the case of `AND` semantics). In that case, the *relativeness* score can be defined as follows:

$$score^{f}_{\vee}(d, E_Q) = \frac{\sum_{e \in E_Q} count(e, d)}{\sum_{e' \in ents(d)} count(e', d)} \cdot \frac{|ents(d) \cap E_Q|}{|E_Q|} \tag{2}$$

where $\frac{|ents(d) \cap E_Q|}{|E_Q|}$ is the percentage of query entities discussed in the document. The score of a document will be 1 if it contains all the query entities and no other entity. This formula favors documents mentioning multiple times many of the query entities.

Now, the probability of a retrieved document $d \in D_Q$ given only the query entities can be defined as:

$$P(d|E_Q) = \frac{score^{f}(d, E_Q)}{\sum_{d' \in D_Q} score^{f}(d', E_Q)} \tag{3}$$

Timeliness

We consider that a time period $t \in T_Q$ is important for the entities in E_Q, if there is a relatively big number of documents in D_Q discussing about these entities during t. For example, a big number of articles about *Nelson Mandela* was published the period 11-13 of February 1990 because in February 11 *Nelson Mandela* was released from prison. Thus, articles published during that period should be promoted since they are probably related to this important event of *Nelson Mandela*'s life.

For the case of `AND` semantics, we define the following importance score of a *time period* $t \in T_Q$:

$$score^{t}_{\wedge}(t) = \frac{|docs(t) \cap D_Q|}{|D_Q|} \tag{4}$$

This scoring formula favors time periods in which there is a big number of documents discussing about the query entities.

For the case of `OR` semantics, in a time period t there may be a big number of documents discussing only for one of the query entities, while in another time period t' there may be a smaller number of documents discussing though for many of the query entities. For also taking into account the number of query entities discussed in documents of a specific time period, we consider the following formula:

$$score^{t}_{\vee}(t) = \frac{|docs(t) \cap D_Q|}{|D_Q|} \cdot N(E_Q, t) \tag{5}$$

where, $N(E_Q, t)$ is the average percentage of query entities discussed in articles of t, i.e.:

$$N(E_Q, t) = \frac{\sum_{d \in docs(t) \cap D_Q} \frac{|ents(d) \cap E_Q|}{|E_Q|}}{|docs(t) \cap D_Q|} \tag{6}$$

Now, the probability of a retrieved document $d \in D_Q$ given only its publication date t_d can be defined as:

$$P(d|t_d) = \frac{score^t(t_d)}{\sum_{d' \in D_Q} score^t(t_{d'})} \tag{7}$$

Relatedness

Entities that are co-mentioned frequently with the query entities in important time periods are probably important. For example, *Apartheid* was an important concept related to *Nelson Mandela* during 1990, thus articles discussing for both *Apartheid* and *Nelson Mandela* should be promoted. However, there may be also some general entities (e.g., *South Africa* in our example) that co-occur with the query entities in almost all documents (independently of the time period). Thus, we should also avoid over-emphasizing documents mentioning such "common" entities.

For the case of `AND` semantics, we consider the following *relatedness* score for an entity $e \in E_D \setminus E_Q$:

$$\begin{aligned} score^r_{\wedge}(e) &= idf_{\wedge}(e) \cdot \sum_{t \in T_Q} (score^t_{\wedge}(t) \cdot \frac{|docs(t) \cap D_Q \cap docs(e)|}{|docs(t) \cap D_Q|}) \\ &= idf_{\wedge}(e) \cdot \sum_{t \in T_Q} \frac{|docs(t) \cap D_Q \cap docs(e)|}{|D_Q|} \end{aligned} \tag{8}$$

where $idf_{\wedge}(e)$ is the inverse document frequency of entity e in the set of documents discussing about the query entities in the entire corpus, which can be defined as follows:

$$idf_{\wedge}(e) = 1 - \frac{|docs(e) \cap (\cap_{e' \in E_Q} docs(e'))|}{|\cap_{e' \in E_Q} docs(e')|} \tag{9}$$

The formula considers the percentage of documents in which the entity co-occurs with the query entities in important time periods.

For the case of `OR` semantics, the above formula does not consider the number of different query entities discussed in documents together with the entity e. To also handle this aspect, we consider the following *relatedness* score for the case of `OR` semantics:

$$\begin{aligned} score^r_{\vee}(e) &= idf_{\vee}(e) \cdot N(E_Q, e) \cdot \sum_{t \in T_Q} (score^t_{\vee}(t) \cdot \frac{|docs(t) \cap D_Q \cap docs(e)|}{|docs(t) \cap D_Q|}) \\ &= idf_{\vee}(e) \cdot N(E_Q, e) \cdot \sum_{t \in T_Q} (N(E_Q, t) \cdot \frac{|docs(t) \cap D_Q \cap docs(e)|}{|D_Q|}) \end{aligned} \tag{10}$$

where $N(E_Q, e)$ is the average percentage of query entities discussed in articles together with entity e, i.e.:

$$N(E_Q, e) = \frac{\sum_{d \in docs(e) \cap D_Q} \frac{|ents(d) \cap E_Q|}{|E_Q|}}{|docs(e) \cap D_Q|} \tag{11}$$

Now the inverse document frequency $idf_{\vee}(e)$ includes documents mentioning at least one of the query entities, i.e.:

$$idf_{\vee}(e) = 1 - \frac{|docs(e) \cap (\cup_{e' \in E_Q} docs(e'))|}{| \cup_{e' \in E_Q} docs(e')|} \tag{12}$$

This formula favors related entities that i) co-occur frequently with many of the query entities, ii) are discussed in documents published in important (for the query entities) time periods.

Now, the probability of a document $d \in D_Q$ given only other entities mentioned in the retrieved documents (E_{D_Q}) can be defined as:

$$P(d|E_{D_Q}) = \frac{\sum_{e \in ents(d) \setminus E_Q} score^r(e)}{\sum_{d' \in D_Q} \sum_{e' \in ents(d') \setminus E_Q} score^r(e')} \tag{13}$$

Joining the Models

We can now combine the different models in a single probability score:

$$P(d|E_Q, t_d, E_{D_Q}) = \frac{P(d|E_Q)P(d|T_Q)P(d|E_{D_Q})}{\sum_{d' \in D_Q} P(d'|E_Q)P(d'|T_Q)P(d'|E_{D_Q})} \tag{14}$$

where the denominator can be ignored as it does not influence the ranking.

4 Stochastic Processing

Our basic idea while performing a Probabilistic Analysis is to dynamically create a graph of results (documents), extracted entities from the results and entities of interest identified by the SPARQL query typed by the user and then using a *PageRank-like* model [**?**] to try to identify the documents which might seem to be of importance to the user.

Defining the Graph

When a query input is given by the user, a we define a graph $G = (V, E)$ in which the node set V consist of the documents obtained as results of the SPARQL query d such that $d \in D_Q$ as well as entities e extracted from these documents. These extracted entities includes the entities of interest $e', e' \in E_Q$. The edge set E consists of edges that are created between:

a) an entity of interest node to an entity node, if the entity node and the entity of interest node are co-mentioned together in at least one returned document by the SPARQL query,
b) a document node to an entity node if the entity finds a mention in the document node.

In the case of `AND` Semantics, there would be edges between every document and all entities of interest nodes since all the result documents will contain a mention of all the entities of interest. There would also be an edge between every entity of interest node to an entity node.

For the case of `OR` Semantics, every document node would have at least an edge connection to an entity of interest node. Likewise, every entity node would be connected to at least one entity of interest node by an edge connection.

Figure **??** shows an example of a small graph for the query *Nelson Mandela* `OR` *Frederik Willem de Klerk* for the time period 10-12 February 1990.

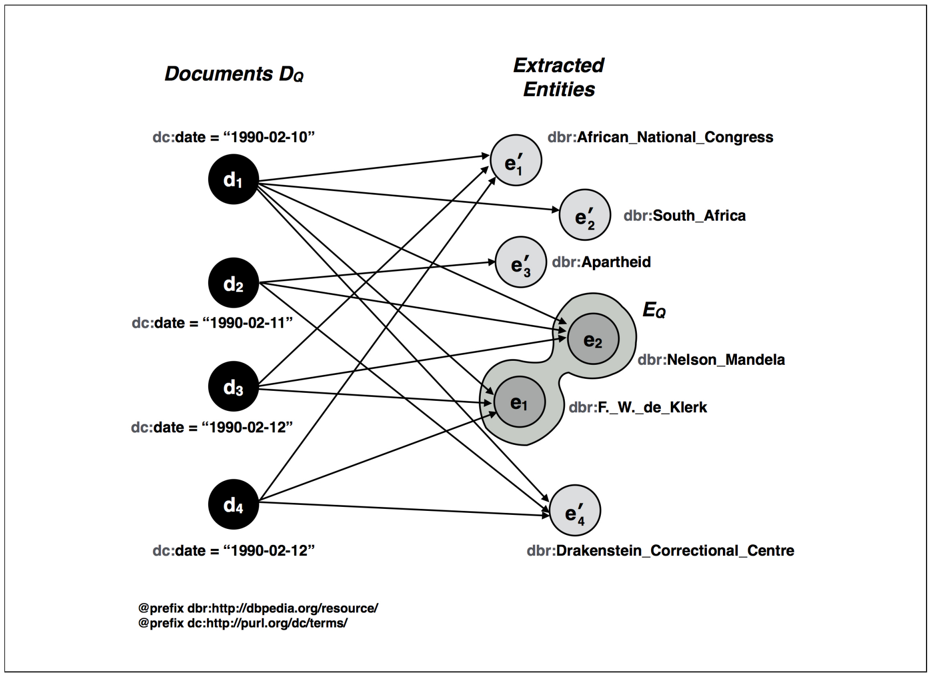

Fig. 5: An example graph for the query *Nelson Mandela* `OR` *Frederik Willem de Klerk*

Creating a Transition Graph

We create a transition graph $G' = (V', E')$ from G where $V' = V$ and E' contains all edges in E with an edge in the opposite direction for each edge in E which has as one of the endpoints a document node, that is, for each $edge(d, e) \in E$ we create $edge(e, d)$. This is based on the notion that if a document *contains* an extracted entity, then the extracted entity is also *contained* in the document.

Describing Random Surfer behaviour

Consider now a Random Surfer beginning at a node in this transition graph and moving to other nodes. In this scenario, the surfer can lie at either document nodes or the extracted entity nodes or the entity of interest nodes.

If the random surfer begins at an entity node e and $e \notin E_Q$, then he can move to a document $d, d \in D_Q$ which contains the entity e.

It can also be the case that the surfer lies at a document node d. In this case, he can transition to an entity e existing/mentioned in the document, i.e., $e \in ents(d)$. Note that in this case e could also be an entity of interest mentioned in the document.

Lastly, the surfer can lie at an entity of interest e'. In this case:
(i) With probability p_1 he moves to a document d, $d \in D_Q$, $e' \in ents(d)$.
(ii) With probability 1-p_1 he moves to an extracted entity e which is co-mentioned together with e' in at least one returned document d, $d \in D_Q$.

Assigning edge weights

The assignment of edge weights needs to be done keeping in mind that the transition probabilities for a surfer from any node must sum up to 1, that is, the weights of all the outgoing edges must sum to 1. We describe weight assignment for each of the cases for nodes the random surfer can lie at and all possible transitions it can make from this node. As described above, the random surfer can only make three types of transitions and in these types we assign edge weights as follows:

Case 1: The surfer lies at an entity e, $e \notin E_Q$ and he moves to a document $d \in D_Q$, i.e., the movement is along the edge $n \to n'$, where $n = e$, and $n' = d, d \in D_Q$.

The weight of the edge $(n \to n')$ in this case $(n = e, n' = d)$ for both `AND` and `OR` semantics can be defined as:

$$weight(e \to d) = \frac{count(e,d)}{\sum_{d' \in docs(e) \cap D_Q}(count(e,d')} \quad (15)$$

The criteria for assigning weight in the first case is based upon the notion that from an entity a surfer is more likely to move to a document that mentions the entity many times.

Case 2: The surfer is at a document d and moves to an entity $e, e \in ents(d)$. The surfer transitions along the edge $(n \to n')$ where $n = d$ and $n' = e$. We perform the weight assignment in this case as follows:

$$weight(d \to e) = \frac{count(e,d)}{\sum_{e' \in ents(d)}(count(e',d))} \quad (16)$$

This means that a surfer is more likely to move from a document node to an entity that finds mention more number of times in the document than other entities.

Case 3a: The walker is at a an entity of interest e' and he moves to a document $d \in D_Q$, i.e., it transitions along the edge $n \rightarrow n'$, where $n = e', e' \in E_Q$ and $n' = d, d \in D_Q$.

The weight of the edge $(n \rightarrow n')$ in this case $(n = e', n' = d)$ for both `AND` and `OR` semantics can be defined as:

$$weight(e' \rightarrow d) = \frac{score^f(d, E_Q) * score^t(t_d)}{\sum_{d' \in docs(e') \cap D_Q}(score^f(d', E_Q) * score^t(t'_d))} \tag{17}$$

Case 3b: The walker lies at an entity of interest e' and moves to a extracted entity e, i.e., it transitions along the edge $(n \rightarrow n')$ where $(n = e', n' = e)$. Considering relatedness, for both `AND` and `OR` Semantics we use the following criteria for assigning weights to the edges:

$$weight(e' \rightarrow e) = \frac{score^r(e)}{\sum_{e'' \in edge(e', e'')}(score^r(e''))} \tag{18}$$

Here, for `AND` and `OR` semantics, we make use of the formulae described for $score^r(e)$ in the previous section accordingly.

Taking into consideration the transition probabilities, the weight from an entity of interest-node e' to a connected node n can be generally defined as:

$$weight(e' \rightarrow n) = \begin{cases} p_1 \cdot \frac{score^f(d, E_Q) * score^t(t_d)}{\sum_{d' \in docs(e') \cap D_Q}(score^f(d', E_Q) * score^t(t'_d))} & n = d, d \in D_Q \\ (1 - p_1) \cdot \frac{score^r(e)}{\sum_{e'' \in edge(e', e'')}(score^r(e''))} & n = e, e \notin E_Q \end{cases} \tag{19}$$

The criteria for assigning weight in the third case is based upon the notion that from an entity of interest a surfer is more likely to move to a document that mentions the entity of interest many times and is published in important time periods (for EoI). It is also likely to move to an extracted entity that it finds frequently co-mentioned in documents together with the entities of interest.

Analyzing the Transition Graph

For a node n, let $in(n)$ be the set of nodes with in-links to n. We define the *PageRank-like* value $r(n)$ as:

$$r(n) = d \cdot Jump(n) + (1 - d) \cdot \sum_{n' \in in(n)} (weight(n \rightarrow n') \cdot r(n')) \tag{20}$$

In this equation d is the *decay factor* or in other words, the probability for the surfer to perform a random jump and with probability *Jump(n)* the surfer jumps to node n. The probability of the walker to transit to node n from a connected node n' is given by $weight(n \rightarrow n')$. Sufficient number of iterations should be performed so as to allow the algorithm to converge. In our experiments we fixed the number of iterations to 30.

Fixing Random Jumps and Tuning

Initially, we start by assigning a PageRank score only to entities of interest. Each entity of interest gets assigned a score equal to 1 divided by the number of entities of interest and all other nodes a score of 0. We fix random jumps such that the surfer can perform random jumps only to entity of interest nodes. Additionally, when performing a random jump, a surfer only jumps equiprobably to any of the entity of interest nodes. We test the algorithm for decay factor values of 0.0, 0.2, 0.5 and 0.8. A value of 0 for the decay factor means that there is no possibility for the surfer to perform a random jump and a value of 1 translates to the surfer only perform jumps from one entity of interest to another entity of interest.

We also vary the p_1 values to see which combination of edge weights leads to the best ranking. A higher p_1 value implies that the edge weights for edges between entity of interest nodes and document nodes increases which might lead to the surfer moving to a highly relevant document from an entity of interest. On the other hand, a lower p_1 values leads to increase in the edge weight between entity of interest nodes and extracted entity nodes which might be beneficial as the surfer would have greater chance to move to a highly related entity node from the entity of interest node and from there it can further transition to a highly relevant document node. The algorithm is tested for different values of p_1 to see which combination leads to the best ranking. A value of 0 for p_1 implies that we the surfer can reach the document nodes from the entity of interest nodes only through the entity nodes thus assigning more importance to highly related entities. Conversely, a p_1 value of 1 would mean that only direct transition is possible from entity of interest node to a document node making documents mentioning the entity of interest many times and published in important time periods more important.

5 Evaluation

5.1 Setup

Our objective is to measure the ranking quality of the results returned by our system and check the usefulness or importance of a document based on its position in the result list and to analyze the ranking models and their combinations.

Due to the absence of a ground truth, we created our own ground truth consisting of 28 queries. The queries comprise of:
- **Query 1-7:** Single-entity queries
- **Query 8-14:** Contain two(or more) entities with AND Semantics

– **Query 15-21:** Contain two(or more) entities with OR Semantics
– **Query 22-28:** Contain entities belonging to a category (OR Semantics)

The queries 1-14 and 15-28 can be broadly grouped as AND Semantics and OR Semantics respectively. The queries were selected from a period of 5 years and many of them are about significant events or prominent entities during that time. For ease of evaluation, we try to restrict the number of results returned to the range of 20 to 60 documents by varying the query time period and for evaluation set granularity to day.

To measure the ranking quality of our ranking system we decided to manually evaluate the results returned by these queries. We built a graded relevance scale of documents using scoring scale of 0 to 3 with the score description as follows:

– **Score 0**: The document has almost nothing to do with the query entities
– **Score 1**: The topic of the document is **not** about the query entities, however the query entities are related to the document context
– **Score 2**: The topic of the document is **not** about the query entities, however the query entities are important for the document context
– **Score 3**: The topic of the document is about the query entities

For measuring Precision, we consider only the documents with manually graded score of 2 or 3 as relevant.
The categorical queries (queries 22-28) use the `SERVICE` operator for querying DBpedia's SPARQL endpoint but not any `OPTIONAL FILTER` operator. The semantic layer was hosted in a Virtuoso server installed in a modest personal computer (MacBook Pro, Intel Core i5, 8GB main memory) and we run the queries in Java 1.8 using Apache Jena 3.1.

5.2 Results

Table 1: NDCG for different ranking models and their combination.

Ranking Model	Normalized Discounted Cumulative Gain (NDCG)							
	AND Semantics				OR Semantics			
	@5	@10	@20	end	@5	@10	@20	end
Random Ranking	0.264	0.352	0.435	0.681	0.271	0.345	0.473	0.676
Relativeness [A]	0.437	0.490	0.595	0.786	0.399	0.434	0.572	0.732
Timeliness [B]	0.274	0.335	0.445	0.685	0.242	0.352	0.488	0.682
Relatedness [C]	0.352	0.434	0.574	0.743	0.457	**0.527**	**0.671**	**0.775**
[A]*[B]	0.490	0.518	0.611	0.796	0.456	0.470	0.601	0.753
[A]*[C]	0.466	0.518	0.618	0.794	0.469	0.497	0.620	0.760
[B]*[C]	0.403	0.471	0.559	0.743	0.486	0.517	0.665	0.772
[A]*[B]*[C]	**0.501**	**0.527**	**0.622**	**0.800**	**0.493**	0.520	0.624	0.771