Back to the Past: Supporting Interpretations of Forgotten Stories by Time-aware Re-Contextualization

Nam Khanh Tran, Andrea Ceroni, Nattiya Kanhabua, and Claudia Niederée

L3S Research Center / Leibniz Universität Hannover, Germany

{ntran, ceroni, kanhabua, niederee}@L3S.de

ABSTRACT

Fully understanding an older news article requires context knowledge from the time of article creation. Finding information about such context is a tedious and time-consuming task, which distracts the reader. Simple contextualization via Wikification is not sufficient here. The retrieved context information has to be time-aware, concise (not full Wiki pages) and focused on the coherence of the article topic. In this paper, we present an approach for time-aware re-contextualization, which takes those requirements into account in order to improve reading experience. For this purpose, we propose (1) different query formulation methods for retrieving contextualization candidates and (2) ranking methods taking into account topical and temporal relevance as well as complementarity with respect to the original text. We evaluate our proposed approaches through extensive experiments using real-world datasets and ground-truth consisting of over 9,400 article/context pairs. To this end, our experimental results show that our approaches retrieve contextualization information for older articles from the New York Times Archive with high precision and outperform baselines significantly.

Categories and Subject Descriptors H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms Algorithms, Experimentation

Keywords Time-aware re-contextualization, Temporal context, Complementarity, News, Interpretation, Wikipedia

1. INTRODUCTION

If we see the 1950's advertisement shown in Figure 1, we might find trouble in understanding the point it makes, or we are outraged about the (not so) implicit message given that women are too weak or too stupid to open a ketchup bottle. At first glance, it seems difficult to imagine how this can be used for advertising a household product. However, if we look into the context information on the right side of the figure, we might understand that the advertisement follows the gender stereotype of a housewife at that time.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WSDM'15, February 2–6, 2015, Shanghai, China.

Copyright 2015 ACM 978-1-4503-3317-7/15/02 ...$15.00.

http://dx.doi.org/10.1145/2684822.2685315.

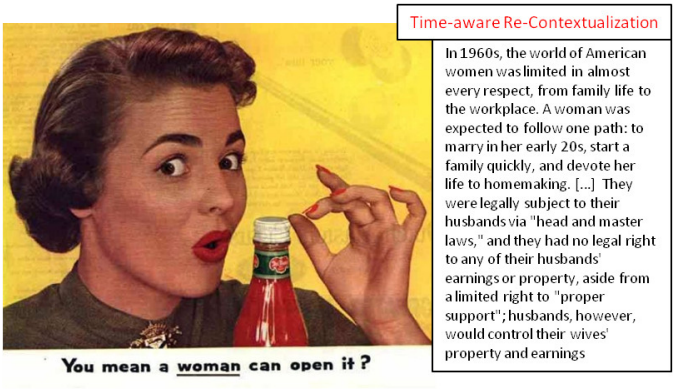

Figure 1: Ketchup advertisement (1953) and its context.

The research challenge addressed in this paper is how such context can be computed for helping in the interpretation of past or forgotten stories, e.g., from a news archive. We call this process *time-aware re-contextualization* [6] or contextualization, for short. The process automatically provides complementing information to a textual document, which reflects required but not expressed context for fully understanding it. Although contextualization might also be necessary due to differences in cultural background or domain expertise, we focus on supporting time-aware interpretation, where a large time-gap between creation and reading time has to be bridged.

The need for dealing with content from the past is not restricted to expert users, such as, journalists, historians or researchers. Due to the growing age of the Web, general Web users are increasingly confronted with the content of different age assuming knowledge of the context at the respective time for its interpretation.

Just adding information, which is related to the entities and concepts mentioned in the text, as it is done in Wikification approaches, for example, [28, 29] or for a domain specific case [17], is not sufficient for many reasons. First, we require a kind of a virtual time-travel, in which - by the information about the past - we are mentally transported into the time of content creation, in our example the US of the 50s. Second, the context information should be digestible in a short time with minimal disruption from the main reading. Therefore, we aim for a contextualization unit granularity, which is considerably smaller than a full Wikipedia page. Finally, contextualization has to coherently consider the specific aspects about entities, concepts or terms, which are relevant in the text under consideration.

Therefore, time-aware re-contextualization, which aims to associate an information item d (such as, a paragraph in a text) with time-aware, concise and coherent context information c for easing its understanding, is a challenging task. Several subgoals of the information search process have to be combined with each other [6]: (1) c has to be relevant for d, (2) c has to complement the information already available in d and the surrounding document, (3) c has to consider the time of creation (or reference) of d, and (4) the context information should be concise to avoid overloading the user.

In [6], we focused on defining the problem and providing a simple proof-of-concept implementation, which relies on manual work for query formulation. The work presented in this paper automates the process of time-aware re-contextualization and provides advanced approaches for retrieval of contextualization candidates and ranking them by taking into consideration complementarity. In more detail, we follow a two-step process. In the first step, we identify contextualization candidates based on contextualization hooks, i.e., the parts of document that require contextualization.[1] For this purpose, we explore and analyze different methods for formulating (generating) queries, which are used for retrieving adequate contextualization candidates from an underlying knowledge source. In the second step, we rank the candidates. Similarly to diversification approaches, (e.g., [38]), this requires balancing two goals: high content-based and temporal relevance for the text to be contextualized, on one hand, and complementarity for providing information that cannot already be found in the text, on the other hand. For our contextualization approach, we use Wikipedia as the knowledge source (because of its world-wide topical and temporal coverage) for contextualizing old stories from news articles.

Our main contributions in this paper can be summarized as follows:

1. We propose effective query formulation methods that take into account the contextualization hooks as well as recall-oriented query performance prediction using a set of novel features for adaptivity to the difficulty of the documents to be contextualized.
2. We present a time-aware ranking method based on learning-to-rank techniques using a novel feature set, and we propose a complementarity computation, which exploits ideas from search result diversification in ranking.
3. Using real-world datasets, we conduct extensive experiments to evaluate our time-aware re-contextualization approach, which achieves high precision and gains considerable improvement over the baselines. For fostering further research on this challenging task, a manually annotated ground-truth is made available.

We start the rest of the paper with a discussion of related work in Section 2. We outline our approach overview in Section 3, and we describe our proposed methods for query formulation and ranking contextualization candidates in Sections 4 and 5, respectively. In Section 6, we describe the experimental settings, followed by a discussion of experimental results in Section 7. Section 8 concludes our paper and presents directions for future work.

[1]Possible contextualization hooks are, for example, entity or concept mentions, and other phrases.

2. RELATED WORK

Basic forms of contextualization have already been suggested in early works (such as [28, 29]). The Wikify! system [28], for example, enables an automated linkage of entity and concept mentions with Wikipedia pages. Meanwhile, a lot of progress has been made in further developing the entity disambiguation step (see e.g. [19]), which is crucial for robust linking of entity mentions to Wikipedia entity pages or entity representations in other knowledge bases, such as, Yago, DBPedia or Freebase. Entity linking, or Entity Disambiguation, detects entity name mentions within text and links them to the corresponding entities in a knowledge base. In contrast to our approach, both Wikification and entity linkage approaches lack two ingredients of time-aware contextualization, (a) they do not take into account the temporal aspect of the text to be enriched and (b) the additional information provided is rather general (e.g., Wikipedia articles about an entity) and not focused to the topical information need resulting from the text under consideration.

In the area of time-aware information retrieval (IR), it has been shown that explicitly modeling the time dimension in ranking can improve the retrieval effectiveness for time-sensitive queries. Basically, there are two types of temporal information particularly useful for time-aware information retrieval: (1) the publication or creation time of a document [20, 22], and (2) temporal expressions mentioned in a document or a query [2]. Aforementioned works address one of two main aspects for temporal relevance, namely, recency ranking [10, 11] or time-dependent ranking [2, 22]. The first aspect takes into account the freshness of web documents, whereas the second aspect considers temporal information needs and the temporal profiles of documents.

Retrieving and processing external information to be added to documents gain increasing interest in the recent years. In [21], for example, news articles are enriched with related predictions – sentences containing temporal references to the future – retrieved from other documents in the same collection. Other works [13, 33, 35] exploit social media (e.g., Twitter) as external sources when processing news articles. In [35], the most *interesting* tweets regarding a given news are selected by formulating the tweet selection as an optimization problem. The objective function, representing how much a tweet set is interesting with respect to a news, takes into account diversity, popularity, authority, and opinions of tweets within the set.

The work in [33] discovers social media utterances that discuss a given news article. Multiple query models are generated from a news article by considering its internal structure, term selection strategies, as well as utterances which explicitly contain links to the article. The different resulting ranked list are then merged through data–fusion techniques. In [13], the authors present a topic modeling approach which jointly exploits news articles and Twitter for event summarization. In order to generate a representative but not redundant summary of an event, complementarity between tweets and news article sentences is assessed by considering both their similarity and their difference. In contrast to those approaches, our work adds the time dimension to the contextualization task. Moreover, we are not looking for more information on the current context, but we try to re-construct the original context of a document.

The contextualization task is also related to the diversification problem in IR [7, 38, 39]. In [38], different metrics

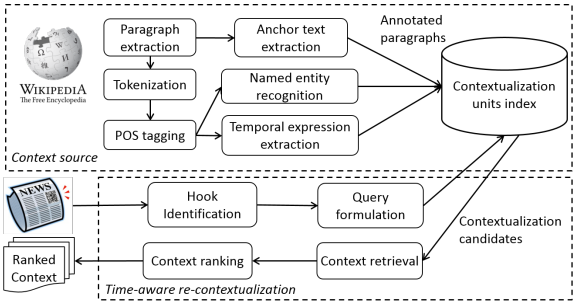

Figure 2: Time-aware re-contextualization approach.

are proposed to measure redundancy in order to investigate the novelty and redundancy of relevant documents in filtering systems. In [7], Clarke et. al. presented a framework for evaluation that systematically rewards novelty and diversity, whereas Zhu et. al. [39] addressed diversification as a learning problem and proposed a novel relational learning-to-rank approach to formulate the task.

In contrast to these studies on analyzing the relation between results to select diverse outcomes for a given query, we mainly focus on the relation between queries (documents in our case) and results (contexts) for finding the ones that are not only topical and temporal relevant, but also complement information already available in the documents.

Automatically formulating queries from text [18] can be done by using tf–idf, mutual information, natural language processing, or machine learning [34, 36, 37]. Assuming the presence of basic metadata and structure for documents, as in [33], some of methods in our paper build queries by exploiting the title and lead paragraph of documents. Similarly to [14], we also explore approaches that assume the availability of manual annotations as seeds for query formulation. The advantage of having such additional information is that the information needs of the users are made explicit, possibly driving to more effective queries. We formulate queries by combining annotations via Query Performance Prediction (QPP) [9], using both pre–retrieval [15] and post–retrieval [5] features. The formers are based only on the query and corpus-based statistics, while the latter also analyze the retrieved list of results. In line with the previous work on time–aware performance predictor [23], we investigate novel features for QPP that explicitly take the temporal dimension into account. Differently from the previously mentioned approaches, which focus on precision metrics, we consider the performances of queries in terms of recall, which have been recently remarked and considered in different information retrieval scenarios [8, 25].

3. APPROACH OVERVIEW

In the general contextualization model underlying our approach we distinguish the information items d to be contextualized and the context source, where the information for the contextualization comes from. Within d a contextualization hook h is an aspect or part of d that requires further information for its time-aware interpretation. The context source is organized into contextualization units cu. In our approach, we have pre-processed a Wikipedia dump as the context source resulting in annotated and indexed Wikipedia paragraphs as contextualization units (see Figure 2). For information items d to be contextualized, we use articles from the New York Times Archive[2] with manually annotated contextualization hooks, i.e., we assume that a reader has marked the places he/she finds difficult to understand.

Starting from the contextualization hooks, the next process is to retrieve a ranked list of contextualization units from the context source. In time-aware re-contextualization the time gap between the creation and reading time of d imposes additional challenges. In our approach, the contextualization process consists of two main steps: (1) formulating queries that are able to retrieve contextualization units, which are good candidates for contextualization; (2) retrieving and ranking the candidates from the context source using the queries from step (1). For step (1) we explore document-based and hook-based query formulation methods and present a procedure that selects good queries based on recall-oriented query performance prediction. For step (2), we employ a retrieval method based on language modeling and re-rank the retrieved contextualization candidates based on a variety of features and a learning to rank approach for ensuring complementarity. The methods developed for steps (1) and (2) are described in more detail in Sections 4 and 5, respectively.

4. QUERY FORMULATION

The goal of the query formulation phase consists of generating a set of queries Q_d for a given document d to retrieve contextualization candidates as input for re-ranking. We explore two families of query formulation methods, one using the document to be contextualized itself as a "generator" of queries (Section 4.1), and the other using contextualization hooks as generators (Sections 4.2 and 4.3). Since some of these methods can generate more than one query from an input document, we will discuss two procedures to merge the ranked result lists in Section 5.1.

4.1 Document-based Query Formulation

The first family of query formulation methods exploits the document content and structure. Similarly to [33], we use three methods to formulate queries from documents: *title*, *lead*, and *title+lead*. *Title* formulates a query consisting of the document title, which is indicative of the main topic of the article. *Lead* uses the lead paragraph of a document, representing a concise summary of the article and including its main actors. *Title+lead*, as a combination of the previous two methods, formulates a query consisting of both the title and the lead paragraph of the document.

Before being performed, all the queries are pre–processed by tokenization, stop-word removal, and stemming. We did not investigate further information extraction approaches for query formulation, since it has been already proven in [33] that the methods described above perform better than more complex information extraction techniques, e.g., keyphrase extraction.

4.2 Basic Hook-based Query Formulation

As already introduced in Section 3, documents in our model are assumed to contain a set of hooks explicitly representing the information needs of the reader or, more precisely, what requires contextualization to be understood and interpreted. The analysis done in [6] showed that contextu-

[2] http://catalog.ldc.upenn.edu/LDC2008T19

alization hooks are not only entity mentions, concept mentions, but also general terms and even short phrases.

We consider two basic hook–based query formulation methods: *all_hooks* and *each_hook*. *All_hooks* includes all the hooks for a document in a single query, representing a tailored perspective of the user's combined information needs for the document. *Each_hook* queries each hook separately, focusing on specific information about single actors, aspects, or sub–topics of the document. The queries generated by these methods are augmented with the title of the document, under the assumption that it is a good representative of the document's topic.

We also experimented with more advanced methods based on identifying hook relationships, for instance considering their co–occurrence in a document collection. However, since these approaches did not perform better than the *all_hooks* method described before, we will not discuss them further.

4.3 Learning to Select Hook-based Queries

Different methods based on ranking and selection of query terms from an initial query might be employed [1, 24, 27], considering the entire set of hooks for a document as the initial query. We explore an adaptive method which formulates queries based on the characteristics of the input document and hooks. Our approach consists of predicting the performances of candidate queries representing subsets of hooks for a given document, ranking them according to the predicted performance, and selecting the top-m of them to be actually performed for the document. The value of m is identified through experiments. In contrast to previous works in query performance prediction, the prediction model is trained on *recall* instead of *precision*. Furthermore, we define novel features for query performance prediction that explicitly take the temporal dimension into account. Finally, our method assesses performances of subsets of query terms (hooks) and can generate more than one query (subsets of hooks).

4.3.1 Candidate Queries

Given a document d and the set of its hooks H_d, we compute its power set $\mathcal{P}(H_d)$ and we create a candidate query for each set of hooks $p \in \mathcal{P}(H_d)$. Again, candidate queries are augmented with the title of the document.

The effort of the computation of features for each element in the power set is not critical in our scenario for two reasons. First, working with short text like news articles limits the number of hooks within the text. Second, the features employed to predict the query performances (Section 4.3.2) are either pre-retrieval measures, which can be computed off–line, or do not require heavy post-retrieval computation.

4.3.2 Features

We measure the performances of each candidate query in terms of its recall because, as already explained, at retrieval phase we are interested in retrieving as much contextualization candidates as possible. In this work we predict query performances with a regression model learned via Support Vector Regression (SVR) [12]. In this model, each learning sample $s = (\boldsymbol{f}_q, r_q)$ consists in a feature vector $\boldsymbol{f}_q$ describing query q (as well as the document it refers to) and its recall r_q, i.e., the label to be predicted. Note that different numbers of top–l results can be used to compute the recall, i.e., the labels, and the choice is discussed in Section 7.

The feature set that we use to represent queries and the document it belongs to are described in the rest of this section. It is composed of novel temporal features for query performance prediction, along with more standard features [4, 16, 30].

Linguistic Features. We compute a family of linguistic features [30] for a query by considering its text and the document it refers to. This results in a set of features both at query and document level: the length of the query, in words; the number of duplicate terms in the query; the number of entities (people, locations, organization, artifacts) in the query; the number of nouns in the query; the number of verbs in the query; the number of hooks in the query; the length of the document's title; the length of the document's lead paragraph; the number of entities in the document (title and lead paragraph); the number of nouns in the documents; the number of verbs in the document; the number of hooks for the document; the number of duplicates in the document.

Document Frequency. The Document Frequency of a hook h represents the percentage of contextualization units in the corpus containing h and it is computed as:

$$df(h) = \log \frac{N_h}{N} \tag{1}$$

where N_h is the number of contextualization units in the corpus containing h and N is the size of the corpus. At document level, we compute the document frequency for every hook of the document the query belongs to, i.e., $df(h)\ \forall h \in H_d$, and then we derive aggregate statistics like average, standard deviation, maximum value, minimum value. Similarly, at query level, we compute $df(h)$ for every hook in the query and we derive the same aggregate statistics as before. In the following, we will refer to average, standard deviation, maximum value, and minimum value simply as *aggregate statistics*.

Temporal Document Frequency. In order to restrict the popularity of a term to a particular time period $T = [t_0 - w; t_0 + w]$, we compute Eq. 1 only for those contextualization units having at least one temporal reference contained in T. This can be done efficiently since contextualization units in our corpus have been annotated with the temporal references mentioned in them. The time period we are interested in is centered around the publication date of the document, i.e., $t_0 = p_d$, and the parameter w determines the width of the interval. After experimenting different values of w, we set $w = 2years$ for our study.

Scope. The scope of a query has been defined in [16] as the percentage of documents (contextualization units in our case) in the corpus that contain at least one query term. Besides the scope of the query itself, we also compute the scope of the document title and the scope of the document hooks H_d when queried together.

Temporal Scope. We define the temporal scope of a query as the percentage of contextualization units in the corpus that contain at least one query term and at least one temporal expression within a given time period. The time period that we consider is the same as the one considered for the computation of temporal document frequency, i.e., a period centered around the publication date of the document and with a temporal window equal to w. Again, we experimented different values of w and we set $w = 2years$.

Relevance. For a given query q, we retrieve the top-k contextualization units and we compute aggregated statis-

tics of their relevance scores given by the underlying retrieval model. The value of k has been empirically set to 100 after experimenting different candidate values. We also computed relevance features at document level, using both document's title and document's hooks as queries.

Temporal Similarity. For a given query q generated from a document d and every retrieved contextualization unit c in its top-k result set (again, $k = 100$), we compute the temporal similarity between q and c and we derive aggregated statistics over the elements in the result set. Temporal similarity between time points t_1 and t_2 is computed through the time-decay function [22]:

$$TSU(t_1, t_2) = \alpha^{\lambda \frac{|t_1 - t_2|}{\mu}} \quad (2)$$

where α and λ are constants, $0 < \alpha < 1$ and $\lambda > 0$, and μ is a unit of time distance. The temporal similarity between a query q and a result c is computed as $\max_{t \in T_c}\{TSU(t, p_d)\}$, where T_c is the set of temporal references mentioned in c and p_d is the publication date of the document where q refers to. This can be done efficiently since temporal references mentioned in contextualization units have been extracted and stored at indexing time.

We also computed temporal similarity features at document level, using both document's title and document's hooks as queries. The computation of the features is the same as the one described above.

5. CONTEXT RANKING

In this section, we describe the methods used in addressing the second part of the contextualization process outlined in Section 3: retrieving and re-ranking context. For the retrieval step, given the queries generated from different query formulation methods described in previous section, we use a retrieval model based on language modeling to create a ranked list of contextualization candidates. Later, learning to select relevant context items is applied to this ranked list.

5.1 Retrieval Model

For the retrieval step, we use query-likelihood language modeling [31] to determine the similarity of a query with the context. In particular, given a query q generated by using one of the methods described in Section 4 for the document d, we compute the likelihood of generating the query q from a language model estimated from a context c with the assumption that query terms are independent.

$$P(c|q) \propto P(c) \prod_{w \in q} P(w|c)^{n(w,q)} \quad (3)$$

where w is a query term in q, $n(w, q)$ is the term frequency of w in q, and $P(w|c)$ is the probability of w estimated using Dirichlet smoothing:

$$P(w|c) = \frac{n(w, c) + \mu P(w)}{\mu + \sum_{w'} n(w', c)} \quad (4)$$

where μ is the smoothing parameter, $P(w)$ is the probability of each term w in the collection.

To combine the rankings produced by each query of a document, we exploited two combining methods namely round-robin, which chooses one result from each ranked list, skipping any result if it has occurred before, and CombSUM, which sums up a result's scores from all ranked lists where it was retrieved. In the experiment, we observed that round-robin method achieves better performance than CombSUM especially in terms of recall, which also reported in [33]. Therefore, we decided to use round-robin method for combining different ranked lists.

5.2 Learning to Rank Context

Once we have obtained a ranked list of contextualization candidates for each document, we turn to context selection (re-ranking) where we need to decide which of the context items are most viable. Our ranking algorithm needs to balance two goals, i.e., high topical and temporal relevance for the document, as well as complementarity for providing additional information. In this work, we use supervised machine learning, that takes as input a set of labeled examples (context to document mappings) and various complementarity features of these examples similar to diversity features [38].

Topic Diversity. This class of features is aimed to compare the dissimilarity between document d and context c on a higher level by representing them using topics. We use latent Dirichlet allocation (LDA) [3] to model a set of implicit topics distribution of the document and context. We define this feature as follows.

$$R_1(c, d) = \sqrt{\sum_{k=1}^{m} (p(z_k|d) - p(z_k|c))^2}$$

where m is the number of topics and z_k is the topic index.

Text Difference. In this case, we represent the document and context as a set of words. The novelty of context c is measured by the number of new words in the smoothed set representation of c. If a word w occurred frequently in context c but less frequently in document d, it is likely that new information not covered by d is covered by c. For computation, a document and its context are represented by a set of informative words (removing stop words and stemming) denoted by $Set(d)$ and $Set(c)$, respectively. We compute the text difference feature as follows.

$$R_2(c, d) = \|Set(c) \cap \overline{Set(d)}\|$$

Entity Difference. The way of computing entity difference is similar to the one for text difference, with the difference that a document and its context are represented by a set of entities. The feature is denoted as $R_3(c, d)$.

Anchor Text Difference. Anchor texts can be regarded as a short summary (i.e., a few words) of the target document and captures what the document is about. This feature can be computed similarly as text and entity features, which is denoted as $R_4(c, d)$. We extract anchor texts using WikiMiner [29] with a confidence threshold γ.

Distributional Similarity. The next feature we use is distribution similarity, which is denoted as $R_5(c, d)$.

$$R_5(c, d) = -KL(\theta_c, \theta_d) = -\sum_{w_i} P(w_i|\theta_c) \log \frac{P(w_i|\theta_d))}{P(w_i|\theta_c)}$$

where θ_d and θ_c are the language models for a document d and its context c, respectively, and are multinomial distributions. We compute θ_d (and similarly for θ_c) using maximum likelihood estimation (MLE) given as:

$$P(w_i|d) = \frac{tf(w_i, d)}{\sum_{w_j} tf(w_j, d)}$$

The problem with using MLE is that if a word never occurs in the document d, the probability $P(w_i|d)$ will be zero; $P(w_i|d) = 0$. Thus, a word in the context c but not in the document d will make $KL(\theta_c|\theta_d) = \infty$. In order to solve this problem, we make use of the Dirichlet smoothing method.

$$P_\lambda(w_i|d) = \frac{tf(w_i, d) + \lambda p(w_i)}{\sum_{w_j}(tf(w_j, d) + \lambda p(w_j))}$$

Geometric Distance. There are several ways to compute geometric distance measure, such as, Manhattan distance and Cosine distance. We leverage Cosine distance because of its robustness to document length.

$$R_6(c, d) = \cos(c, d) = \frac{\sum_{k=1}^{n} w_k(c) w_k(d)}{\|d\| \ \|c\|}$$

In our experiment, we used each unique word as one dimension and the *tf.idf* score as the weight of each dimension.

Relevance and temporal features. In order to retrieve high topical and temporal relevant contextualization candidates for the document, we consider also relevance and temporal features. For the former one, we exploit the retrieval scores of context returned by our retrieval model. For the later one, we apply temporal similarity measurement, i.e., TSU which is described in the previous section.

6. EXPERIMENTAL SETUP

Document Collections. In our experiments, we used the New York Times Annotated Corpus, which contains 1.8 million documents from January 1987 to June 2007, as the document collection to be contextualized. For context source, we employed Wikipedia because it is considered the largest and most up-to-date online encyclopedia covering a wide temporal range of general and specific knowledge. We obtained the Wikipedia dump of February 4, 2013 and considered *paragraphs* as contextualization units. In this particular snapshot, we obtain 4,414,920 Wikipedia articles that contain 25,708,539 paragraphs. For each paragraph, we used Stanford CoreNLP [26] for tokenization, entity annotation and temporal expression extraction. In addition, *anchor* texts found in the paragraph hyperlinks are also extracted. We used Apache Solr[3] to index the annotated paragraphs.

Ground-truth Dataset. In order to obtain ground-truth dataset (both for training and evaluation), we manually selected a set of 51 articles that spanned a wide range of topics (business, technology, education, science, politics, and sports) focusing on the older ones (29 articles published in 1987, 2 articles in 1988, 6 articles in 1990, 7 articles in 1991, and 7 articles in 1992) and recruited six human annotators to manually annotate those articles. The annotators were presented with an annotation interface with which they can evaluate article/context pairs (relevant or non-relevant). The annotation guidelines specified that the annotators should assign relevance to the context that contains *additional information* which complements the information in the article and does provide a good answer to (at least) one of the questions they think up when reading the article. For each article, we retrieved up to 20 contextualization candidates with each query formulation method and removed duplicates afterwards. In total, our annotation dataset consists of 9,464 article/context pairs, where the annotators evaluated 26.9 relevant context per article on average. To foster further research on this challenging task, our ground-truth dataset is publicly available.[4] We measured the inter-annotator agreement using Cohen's kappa statistic. We averaged the pairwise kappa values of all possible combinations of annotators that had overlapping candidates they had annotated and we obtained a fair agreement of κ_c = 0.37 given the high complexity of this contextualization task, which includes objectivity and subjectivity.

Parameter Settings. For query performance prediction, the regression model described in Section 4.3.2 was built by using the Support Vector Regression implementation of LibSVM[5]. In particular, we trained a n–SVR model with Gaussian Kernel through 10–fold cross validation. The open parameters were tuned via grid search to $C = 3$, $\gamma = 0.5$ and $\nu = 0.75$. Linguistic features were extracted using Stanford CoreNLP [26].

For re-ranking context, we performed 5-fold cross validation at document level. We reported scores averaged over all testing folds. We conducted experiments using several machine learning algorithms to confirm the robustness of our approach, i.e., it does not depend on any specific algorithm. In this paper, we employed Random forests (RF), RankBoost (RB) and AdaRank that are implemented in RankLib.[6] In order to compute topic-based feature, we employed the topic modeling tool Mallet[7] by specifying the number of topics to 100, for this task. In addition, we set the confidence threshold to $\gamma = 0.3$ for extracting anchor texts using WikiMiner. For smoothing, we set $\mu = 2000$ and $\lambda p(w_i) = 0.5$.

For computing temporal similarity feature, we set $\lambda = 0.25$, $\alpha = 0.5$, and $\mu = 2years$ in our experiments. We also observed that changing those parameters did not affect the correlation capabilities of the feature.

Evaluation Metrics. The evaluation metrics, we considered precision at rank 1, 3, 10 (P@1, P@3, P@10 respectively), recall, and mean average precision (MAP). These measures provide a short summary of quality of the retrieved context. In our experiment, a context is considered relevant if it is marked as relevance by an annotator, otherwise we consider it as non-relevance. We used the top-20 returned context for evaluation because it is not expected that readers consider more than 20 contextualization units. Statistical significance was performed using a two-tailed paired t-test and is marked as $^{\blacktriangle}$ and $^{\triangle}$ for a significant improvement (with $p < 0.01$ and $p < 0.05$, respectively), and significant decrease with $^{\blacktriangledown}$ and $^{\triangledown}$ (for $p < 0.01$ and $p < 0.05$, respectively).

Baselines. For comparing to our approach, we considered three following competitive baselines.

Milne and Witten (M&W). The method proposed by Milne and Witten [29] which represents the state-of-the-art in automatic linking approaches. We use the algorithm and best-performing settings as described in [29]. In order to apply this method for our task, we consider all paragraphs of all linked pages as a candidate set.

Language Model (LM). The standard query-likelihood language model is used for the initial retrieval as described in Section 5 which provides the top retrieved documents as a candidate set for the contextualization task.

[3] https://lucene.apache.org/solr/

[4] http://www.l3s.de/~ntran/contextualization/

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[6] http://sourceforge.net/p/lemur/wiki/RankLib/

[7] http://mallet.cs.umass.edu/topics.php

Time-aware Language Model (LM-T). Since we aimed at adding context to past stories, the temporal dimension is important. We selected a state-of-the-art time-aware ranking method, which has been shown very effective for answering temporal queries, as our third baseline. It assumes the textual and temporal part of the document d are generated independently from the corresponding parts of the context c, yielding

$$P(d|c) = P(d_{text}|c_{text}) \times P(d_{time}|c_{time}) \quad (5)$$

where d_{time} is the document's publication date, c_{time} is the set of temporal expressions in the context c.

The first factor $P(d_{text}|c_{text})$ can be computed by Eq. 3 and Eq. 4. The second factor in (5) is estimated, based on a simplified variant of [2], as

$$P(d_{time}|c_{time}) = \frac{1}{\mid c_{time} \mid} \sum_{t \in c_{time}} P(d_{time}|t) \quad (6)$$

If the document has zero probability of being generated from the context, Jelinek-Mercer smoothing is employed, and we estimate probability of generating the document's publication date from context c as

$$P(d_{time}|c_{time}) = (1-\lambda)\frac{1}{\mid C_{time} \mid} \sum_{t \in C_{time}} P(d_{time}|t) + \lambda \frac{1}{\mid c_{time} \mid} \sum_{t \in c_{time}} P(d_{time}|t) \quad (7)$$

where $\lambda \in [0,1]$ is a tunable mixing parameter which is set to $\lambda = 0.5$ in our experiment (changing this parameter does not affect our results), and C_{time} refers the temporal part of the context collection treated as a single context and $P(d_{time}|t)$ is estimated by using time-decay function, i.e., TSU computed as in Eq. 2.

7. RESULTS AND DISCUSSION

7.1 Query Formulation

We evaluate and compare the performances of the different query formulation methods described in Section 4, focusing on recall metric. The results reported in the rest of this section are averaged over the 51 documents in our dataset.

In order to fairly evaluate and compare the recall capabilities of the different methods, which can generate different numbers of queries, we allow each method to retrieve the same number of results k. The choice of the method that we used to create a single result set of k elements from different ranked lists have been discussed in Section 5.1.

7.1.1 Prediction Performances

The query formulation method described in Section 4.3 is based on predicting the performances (recall in our case) of candidate queries, ranking them according to the prediction, and then using the top-m queries to retrieve results. Thus, the quality of the query performance prediction itself has to be evaluated before assessing and comparing the performances of the whole query formulation method.

The regression model has been trained via 10–fold cross validation, and the results reported hereafter have been averaged over the 10 folds. The Correlation Coefficient is equal to 0.973, the Root Mean Squared Error equals to 0.056, and the Mean Absolute Error equals to 0.037. The low error values and high correlation value, if compared with the performances in predicting query precision reported in previous works (e.g. [5, 32]), show that the recall of queries in our task can be predicted quite accurately by using the features described in Section 4.3.2.

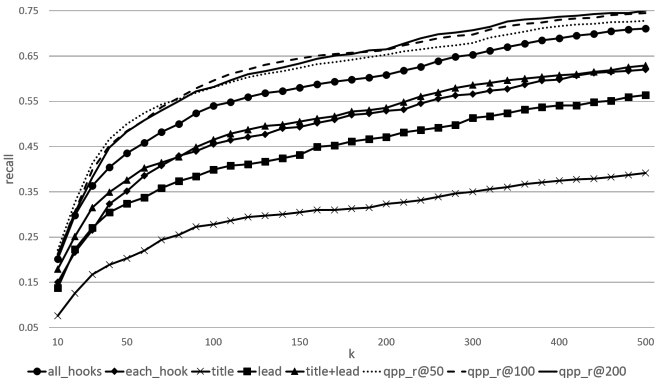

Figure 3: Recall curves of document-based and hook-based methods.

Feature Analysis. In order to analyze which are the most important features in our model, we identified the top–10 features according to their absolute correlation coefficient. Referring to Section 4.3.2, these are: *max query relevance*, *number of hooks in document*, *min document's hooks df*, *max document's hooks temporal df*, *document's hooks scope*, *avg query temporal similarity*, *document's title temporal scope*, *std query relevance*, *avg document's title temporal similarity*, and *std query temporal similarity*. The presence of temporal document frequency, temporal similarity, and temporal scope shows that the temporal features that we defined play an important role in the model. We can also note that both query–level and document–level features are important, since the set is made of 4 features from the former and 6 features from the latter class. Finally, there is only one linguistic feature in the set, namely the number of hooks in the document, confirming that this class of features alone does not correlate well with query performances [4].

7.1.2 Comparison of Query Formulation Methods

In this section, we compare recall values for the document–based methods (*title*, *lead*, *title+lead*), the basic hook–based methods (*each_hook*, *all_hooks*), as well as the method based on query performance prediction, hereafter called *qpp*. For the latter method, we report the performances achieved when using prediction models trained with different labels: we experimented with different l values, namely $l = 50, 100$ and 200, for the computation of the recall at l to be used as label. These three methods will be called *qpp_r@50*, *qpp_r@100* and *qpp_r@200*, respectively, in the rest of the experiments. Note that each *qpp* method considered here uses the top-2 queries, according to their predicted performances, to retrieve the results. The choice of selecting $m = 2$ queries will be explained in more detail in Section 7.1.3.

The recall curves of the different methods, for different values of top–k results, are shown in Figure 3. The curves of *title* and *lead* are the lowest ones, while their combination (*title + lead*) becomes comparable with *each_hook*. Querying using all the hooks of a document together, i.e., *all_hooks*, exposes higher recall values than all the aforementioned methods, showing that performing hook–based

	R@50		R@100		R@200	
	qpp	*all_hooks*	*qpp*	*all_hooks*	*qpp*	*all_hooks*
easy	0.6208	0.5666	0.7361	0.6969	0.7951	0.7686
hard	0.3837	0.3094	0.4606	0.3892	0.5391	0.4550

Table 1: Recall of *all_hooks* and *qpp* methods over different classes of documents grouped by their retrieval difficulty.

queries does lead to better performances in terms of recall with respect to document–based methods. The difference in performances between *each_hook* and *all_hooks* is due to the fact that querying all the hooks together prefers contextualization candidates that contain many hooks. These are potentially more relevant, as they refer to different aspects (hooks) of the same document. Regarding the *qpp* methods, for $k > 20 - 30$, the recall values achieved are between 3% and 7% higher than the ones obtained by *all_hooks*. For larger values of k, e.g. $k > 400$, the difference between the *qpp* methods and *all_hooks* reduces because the prediction models used by the *qpp* methods have been optimized for lower values of k (recall that $l = 50, 100, 200$). This means that, if the number of k results to be retrieved for the re-ranking phase is known and fixed in advance, this information can be exploited early in the training of the query performance prediction model by setting $l = k$, leading to higher recall values for that particular k.

Another comparative analysis between *qpp* methods and *all_hooks* can be done by categorizing the documents according to their *difficulty*, which we define in terms of the amount of relevant context that can be retrieved for a given document. This means that difficult documents are those for which few relevant context can be retrieved, before the re-ranking phase. We categorize documents in *easy* and *hard* with respect to the *all_hooks* method, since it represents a baseline in this comparative analysis with *qpp* methods.

The splitting of the documents in easy and hard was performed by considering the recall at $k = 200$ achieved by *all_hooks* for the different documents. Since the recall values associated to the different documents exhibited a uniform distribution, we split the document set in two equal parts, one representing easy documents and the other representing hard documents.

Table 1 shows the performances of *qpp_r*@50, *qpp_r*@100, and *qpp_r*@200 compared to the ones of *all_hooks* for the different categories of difficulty. The comparison between each *qpp* method and *all_hooks* is done considering the recall at those k values used to train the prediction model (i.e. $k = l$, $l = 50, 100, 200$). Besides *qpp_r*@50, *qpp_r*@100, and *qpp_r*@200 are on average better than *all_hooks* both for easy and hard documents, their improvements are greater for hard documents. In case of *qpp_r*@100, for instance, the relative improvement with respect to the recall value achieved by *all_hooks* is 5.6% for easy documents and 18.3% for hard documents. We believe that the capability of getting higher recall improvements for documents whose relevant context units are difficult to retrieve is a considerable characteristic for the *qpp* methods.

As a conclusion, in this section we proved that exploiting hooks in query formulation is more effective, in terms of recall, than document–based query formulation methods. Moreover, we showed that learning to select candidate hook-based queries can be better, again in terms of recall, than the basic hook–based query formulation methods.

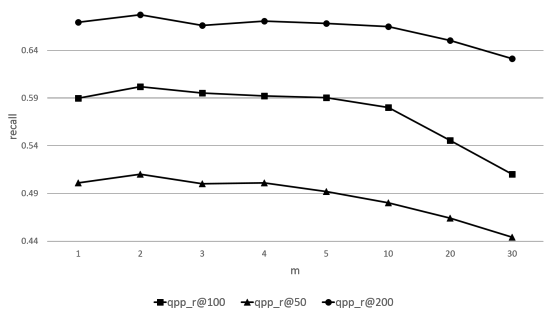

Figure 4: Recall values of *qpp_r*@50, *qpp_r*@100, and *qpp_r*@200 by varying the number of top–m queries.

7.1.3 Number of Queries

The number of top ranked queries that *qpp* methods perform is an open parameter, which we tuned via an empirical analysis observing the recall performances when selecting different numbers of top–m ranked queries. Recall that, for sake of fair comparison, we allow each method to pick the same number of results k from the result lists retrieved by the queries that it generated for a given document. This means that increasing the number of queries to be selected and performed does not necessarily lead to higher recall.

Figure 4 shows the recall values achieved by *qpp_r*@50, *qpp_r*@100 and *qpp_r*@200 (computed at top–50, top–100 and top–200 results, respectively) for different numbers of top–m selected queries. A common trend over the different curves can be observed that they stay quite stable for small values of m, exhibiting a little peak for $m = 2$, and then they decrease for increasing values of m. After observing this behavior, we decided to fix the number of performed queries to $m = 2$.

7.2 Context Ranking

In this subsection, we report the retrieval performances of different query formulation methods and analyze the effectiveness of our context ranking methods trained by using different machine learning algorithms. Firstly, we investigate the performance of the standard, well-known Wikification technique, i.e., the M&W method, in retrieving contextualization candidates. Our experiment considers all paragraphs of all linked pages as candidates. This method achieves the low recall value of 0.229, which indicates that current semantic linking approaches are not appropriate for the contextualization task.

Table 2 shows the results of different query formulation methods. The first group (top) reports results for candidate retrieval based on document-based query models in which the best performing model is *title + lead* that uses content from the article's title and lead paragraph. Turning into models derived from contextualization hooks, Table 2 shows that the *qpp_r@100* model performs the best among all hook-based query models and significantly improves over *title + lead* on all metrics.

Similar to the previous experiment, Table 3 reports the results of *all_hooks* and *qpp_@100* retrieval baselines on a subset of difficult documents (here recall is computed on top-20 candidates). On this subset, *qpp_r@100* also shows

	P@1	P@3	P@10	MAP	Recall
Document-based query models					
title	0.2156	0.1895	0.1745	0.2446	0.1211
lead	0.4902$^{\blacktriangle}$	0.4641$^{\blacktriangle}$	0.3333$^{\blacktriangle}$	0.4908$^{\blacktriangle}$	0.2603$^{\blacktriangle}$
title + lead	0.5294$^{\blacktriangle}$	0.4705$^{\blacktriangle}$	0.3901$^{\blacktriangle}$	0.5161$^{\blacktriangle}$	0.2723$^{\blacktriangle}$
Basic hook-based query models					
each_hook	0.3333	0.3464	0.2745	0.4003	0.1969
all_hooks	0.5490	0.5098	0.4137	0.5640	0.2979
Query performance prediction model					
qpp_r@100	**0.5882**	**0.5490**$^{\blacktriangle}$	**0.4529**$^{\blacktriangle}$	**0.5802**$^{\blacktriangle}$	**0.3097**$^{\blacktriangle}$

Table 2: Retrieval performance of document-based and hook-based query models. The significance test is compared with Row 1 (within the first group) and Row 3 (for the second and third groups).

	P@1	P@3	P@10	MAP	Recall
all_hooks	0.5000	0.3462	0.2885	0.4487	0.2217
qpp_r@100	0.5000	**0.4743**$^{\triangle}$	**0.3730**$^{\triangle}$	**0.5048**$^{\triangle}$	**0.2357**

Table 3: Retrieval performance of *all_hooks* and *qpp_@100* on a set of difficult documents.

significant improvement over *all_hooks* in terms of precision. In short, the results on different query formulation methods indicate that using hook-based approaches outperforms the document-based approach that based on merely article internal structure. Using the query performance prediction method obtains the highest performance on all metrics, followed by *all_hooks*.

We now present the results of our re-ranking approach when using a set of innovative complementarity features to further improve performances of the context ranking step, especially in terms of precision. We select *title + lead* for the document-based approach and *all_hooks, qpp_r@100* for the hook-based approach.

	P@1	P@3	P@10	MAP	Recall
title + lead					
LM	0.5294	0.4705	0.3901	0.5161	0.2723
RF	**0.7672**$^{\blacktriangle}$	0.5757$^{\triangle}$	**0.4909**$^{\blacktriangle}$	**0.6170**$^{\blacktriangle}$	**0.3522**$^{\blacktriangle}$
RB	0.6036	**0.5945**$^{\triangle}$	0.4694$^{\blacktriangle}$	0.5945	0.3417$^{\blacktriangle}$
AdaRank	0.6254	0.5406	0.4143	0.5457	0.3249
all_hooks					
LM	0.5490	0.5098	0.4137	0.5640	0.2979
RF	**0.8272**$^{\blacktriangle}$	**0.6630**$^{\blacktriangle}$	**0.5014** $^{\blacktriangle}$	**0.6427**$^{\triangle}$	0.3611$^{\blacktriangle}$
RB	0.7855$^{\blacktriangle}$	0.6593$^{\blacktriangle}$	0.5009$^{\blacktriangle}$	0.6475$^{\triangle}$	**0.3637**$^{\blacktriangle}$
AdaRank	0.6472	0.5836	0.4687	0.6034	0.3372$^{\triangle}$
qpp_r@100					
LM	0.5882	0.5490	0.4529	0.5802	0.3097
RF	**0.8054**$^{\blacktriangle}$	**0.6993**$^{\blacktriangle}$	0.5140$^{\blacktriangle}$	0.6498$^{\blacktriangle}$	**0.3951**$^{\blacktriangle}$
RB	0.7218	0.6915$^{\blacktriangle}$	**0.5300**$^{\blacktriangle}$	**0.6632**$^{\blacktriangle}$	0.3792$^{\blacktriangle}$
AdaRank	0.6072	0.6139	0.4895	0.6109	0.3479$^{\blacktriangle}$

Table 4: Retrieval performance of different machine-learned ranking methods compared to the best performing retrieval baselines.

The first (top) group in Table 4 shows the results when applying machine learning to *title + lead* retrieval baseline. All three algorithms are able to improve precision at rank k, MAP and Recall. Random forest (RF) and RankBoost (RB) obtain significant improvement where RF achieves the highest scores on most metrics, except precision at rank 3 where RB is the best. The second (middle) group reports the results of *all_hooks* retrieval baseline, augmented by the re-ranking step. In this case, RF and RB are again able to significantly improve over *all_hooks* on all metrics while AdaRank is also performing significantly better than *all_hooks* in terms of recall. Among three algorithms, RF achieves the highest results, except for recall. Similarly, all three machine learning algorithms perform significantly better than the *qpp_@100* retrieval baseline. Again, in this case RF obtain the highest performances, closely followed by RB.

In order to compare our approach to time-aware language model which takes into account temporal information, we use the queries derived from query performance prediction method, i.e., *qpp_@100* that obtain the highest results among our query formulation methods. Table 5 shows that using time-aware language models is not efficient in our case. This is possibly due to that lots of relevant context (paragraphs in our case) do not have any temporal information as shown in Figure 5. Consequently, these candidates are ranked low (e.g., higher than 20) in the ranked list returned by LM-T. This result indicates that purely using the time dimension in context retrieval is not sufficient in the contextualization task. It also confirms the importance of complementarity that is used in our re-ranking step.

qpp_@100	P@1	P@3	P@10	MAP	Recall
LM-T	0.5882	0.4967	0.4176	0.5446	0.2796
LM	0.5882	0.5490	0.4529	0.5802	0.3097$^{\triangle}$
RF	**0.8054**$^{\triangle}$	**0.6993**$^{\blacktriangle}$	**0.5140**$^{\blacktriangle}$	**0.6498**$^{\blacktriangle}$	**0.3951**$^{\blacktriangle}$

Table 5: Retrieval performance of our proposed ranking method and the state-of-the-art time-aware language modeling approach. The significance test is compared against LM-T.

8. CONCLUSIONS

In this paper we have presented an approach for the novel and challenging task of time-aware re-contextualization of content with a gap between creation and reading time. We have shown that our approach can compute relevant and complementing contextualization information with high precision. In the experiments, hook-based query formulation methods have outperformed document-based ones supporting the validity of our contextualization model, and the predominance of query formulation methods relying on several hooks shows the importance of comprehensive contextualization approaches that go beyond the consideration of individual hooks. Furthermore, our experiments have confirmed that complementarity, which is used in the re-ranking step, plays an important role in contextualization.

Although the results achieved in this paper are promising, the task of time-aware re-contextualization still requires further investigation. As a future work, we have planned to further improve our methods and to investigate into strategies for automatic hook identification. Furthermore, we are planning to combine our re-ranking approach with a diver-

News article - *Maj. Gen. Richard V. Secord, a main organizer of the Iran arms sales and the contra supply operation, testified today that he had been told that President Reagan had been informed that proceeds from the sales to Iran had been diverted to the Nicaraguan rebels.*

Context - *Speaking of the Iran-Contra affair, a Reagan administration scandal that involved the diverting of funds being shipped to Iran to the contras in Nicaragua, Reagan says, "None of the arms we'd shipped to Iran had gone to the terrorists who had kidnapped our citizens." Of the scandal, Reagan writes, " and, I presume, knew how deeply I felt about the need for the contras' survival as a democratic resistance force in Nicaragua. Perhaps that knowledge... led them to support the contras secretly and saw no reason to report this to me." He also says of himself, "As president, I was at the helm, so I am the one who is ultimately responsible."*

Figure 5: Example of contextualization candidate for a given document with no explicit temporal information.

sification approach, such that we can compile the retrieved contextualization information into concise contextualization sets with little overlap between the items. Finally, we also plan to look into the personalization of contextualization approaches taking into account individual differences in contextualization needs.

Acknowledgments The work was partially funded by the European Commission for the FP7 project ForgetIT under grant No. 600826 and the German Federal Ministry of Education and Research (BMBF) for the project *"Gute Arbeit" nach dem Boom (Re-SozIT)* (01UG1249C).

9. REFERENCES

[1] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR '08*, 2008.

[2] K. Berberich, S. J. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *ECIR '10*, 2010.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 2003.

[4] D. Carmel and O. Kurland. Query performance prediction for ir. In *SIGIR '12*, 2012.

[5] D. Carmel and E. Yom-Tov. Estimating the query difficulty for information retrieval. In *SIGIR '10*, 2010.

[6] A. Ceroni, N. K. Tran, N. Kanhabua, and C. Niederée. Bridging temporal context gaps using time-aware re-contextualization. In *SIGIR '14*, 2014.

[7] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*, 2008.

[8] G. V. Cormack and M. R. Grossman. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *SIGIR '14*, 2014.

[9] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR '02*, 2002.

[10] N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *SIGIR '11*, 2011.

[11] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *WSDM '10*, 2010.

[12] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *NIPS '97*, 1997.

[13] W. Gao, P. Li, and K. Darwish. Joint topic modeling for event summarization across news and social media streams. In *CIKM '12*, 2012.

[14] G. Golovchinsky, M. N. Price, and B. N. Schilit. From reading to retrieval: Freeform ink annotations as queries. In *SIGIR '99*, 1999.

[15] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *CIKM '08*, 2008.

[16] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *SPIRE '04*, 2004.

[17] J. He, M. de Rijke, M. Sevenster, R. van Ommering, and Y. Qian. Generating links to background knowledge: a case study using narrative radiology reports. In *CIKM '11*, 2011.

[18] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. In *WWW '03*, 2003.

[19] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *EMNLP '11*, 2011.

[20] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 2007.

[21] N. Kanhabua, R. Blanco, and M. Matthews. Ranking related news predictions. In *SIGIR 11*, 2011.

[22] N. Kanhabua and K. Nørvåg. Determining time of queries for re-ranking search results. In *ECDL '10*, 2010.

[23] N. Kanhabua and K. Nørvåg. Time-based query performance predictors. In *SIGIR '11*, 2011.

[24] C.-J. Lee, R.-C. Chen, S.-H. Kao, and P.-J. Cheng. A term dependency-based approach for query terms ranking. In *CIKM '09*, 2009.

[25] C. Li, Y. Wang, P. Resnick, and Q. Mei. Req-rec: High recall retrieval with query pooling and interactive classification. In *SIGIR '14*, 2014.

[26] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL '14*, 2014.

[27] K. T. Maxwell and W. B. Croft. Compact query term selection using topically related text. In *SIGIR '13*, 2013.

[28] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM '07*, 2007.

[29] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM '08*, 2008.

[30] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty. In *SIGIR '05*, 2005.

[31] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, 1998.

[32] F. Raiber and O. Kurland. Query-performance prediction: Setting the expectations straight. In *SIGIR '14*, 2014.

[33] M. Tsagkias, M. de Rijke, and W. Weerkamp. Linking online news and social media. In *WSDM '11*, 2011.

[34] P. D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2000.

[35] T. Štajner, B. Thomee, A.-M. Popescu, M. Pennacchiotti, and A. Jaimes. Automatic selection of social media responses to news. In *KDD '13*, 2013.

[36] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *DL '99*, 1999.

[37] Y. Yang, N. Bansal, W. Dakka, P. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *WSDM '09*, 2009.

[38] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR '02*, 2002.

[39] Y. Zhu, Y. Lan, J. Guo, X. Cheng, and S. Niu. Learning for search result diversification. In *SIGIR '14*, 2014.