Exploiting Time-based Synonyms in Searching Document Archives

Nattiya Kanhabua
Dept. of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
nattiya@idi.ntnu.no

Kjetil Nørvåg
Dept. of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
noervaag@idi.ntnu.no

ABSTRACT

Query expansion of named entities can be employed in order to increase the retrieval effectiveness. A peculiarity of named entities compared to other vocabulary terms is that they are very dynamic in appearance, and synonym relationships between terms change with time. In this paper, we present an approach to extracting synonyms of named entities over time from the whole history of Wikipedia. In addition, we will use their temporal patterns as a feature in ranking and classifying them into two types, i.e., time-independent or time-dependent. Time-independent synonyms are invariant to time, while time-dependent synonyms are relevant to a particular time period, i.e., the synonym relationships change over time. Further, we describe how to make use of both types of synonyms to increase the retrieval effectiveness, i.e., query expansion with time-independent synonyms for an ordinary search, and query expansion with time-dependent synonyms for a search wrt. temporal criteria. Finally, through an evaluation based on TREC collections, we demonstrate how retrieval performance of queries consisting of named entities can be improved using our approach.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Measurement

Keywords

Temporal Search, Synonym Detection, Query Expansion

1. INTRODUCTION

In recent years, an enormous amount of information has been stored in the form of digital documents. Examples of easily available resources include web pages stored by search engines, and harvested web pages stored by web archives, e.g., the Internet Archive, national libraries, and news archives, such as The Times Online.

Much of the content in the resources mentioned is strongly time-dependent. As have been observed by a number of researchers [2, 13], extending keywords with a creation or update date of documents (called temporal criteria) can increase precision of search. In that way, a system narrows down a set of results by retrieving documents according to both keywords and temporal criteria, e.g., temporal text-containment search [2, 13]. Two ways of obtaining temporal criteria relevant to a query are 1) having them provided by users [13], or 2) determined by the system [8].

One way of increasing recall is to perform query expansion. A particular case of query expansion is when search terms are *named entities* (i.e., name of people, organizations, locations, etc.) which constitute a major fraction of queries [4, 14]. In this case, recall can be increased by also searching for synonyms[1] of the named entities. A problem of query expansion using synonyms is the effect of rapidly changing synonyms of named entities over time, e.g., changes of roles or alterations of names. To illustrate the problem, we give as examples two search scenarios.

First, a student studying the history of the Roman Catholic wants to know about the Pope Benedict XVI during the years before he became the Pope (i.e. before 2005). Using only the query "Pope Benedict XVI" and temporal criteria "before 2005" is not sufficient to retrieve documents about "Joseph Alois Ratzinger", which is the birth name of the current Pope.

Second, a journalist wants to search for information about the past career of Hillary Rodham Clinton before becoming the 67th United States Secretary of State in January 2009. When searching with the query "Hillary R. Clinton" and temporal criteria "before 2008", documents about "United States Senator from New York" and "First Lady of the United States" are also relevant as her roles during the years before 2008.

The above examples indicate an inability of retrieving relevant documents composed of the synonyms of query terms in the past. This can be considered as *semantic gaps* in searching document archives, i.e., a lack of knowledge about a query and its synonyms, which are semantically equivalent/related to a query wrt. time. We denote those synonyms as *time-dependent synonyms*.

A peculiarity of named entities compared to other vocabulary terms is that they are very dynamic in appearance, every day new named entities are indexed and searched for, and at the same time existing named entities disappear from interest. This implies that if query expansion techniques for named entities should have good performance, time has to be taken into account, and continuously evolving named entities and synonyms have to be maintained.

In this paper, we describe an approach to automatically creating entity-synonym relationships based on the contents of Wikipedia.

[1]In general, synonyms are different words with very similar meanings. However, in this paper, synonyms are words used as another name for an entity.

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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
JCDL'10, June 21–25, 2010, Gold Coast, Queensland, Australia.
Copyright 2010 ACM 978-1-4503-0085-8/10/06 ...$10.00.

Evolving relationships are detected using the most current version of Wikipedia, while relationships for particular time in the past are discovered through the use of snapshots of previous Wikipedia versions. In this way, we can provide a source of time-based entity-synonym relationships from 2001 until today, and using our approach also future relationships with new named entities can be discovered simply by processing Wikipedia as new contents are added. Further, we employ the New York Times Annotated corpus in order to extend the covered time range as well as improve the accuracy of time of synonyms.

The main contributions of this paper are: 1) formal models for Wikipedia viewed as a temporal resource and for classification of time-based synonyms, 2) an approach to discovering time-based synonyms using Wikipedia and improving the time of synonyms, 3) a study on how to perform query expansion using time-based synonyms, and 4) an extensive evaluation of our approaches to extracting and improving time of synonyms, as well as of query expansion using time-based synonyms.

The organization of the rest of the paper is as follows. In Section 2, we give an overview of related work. In Section 3, we briefly describe the assumed document model and Wikipedia features. In Section 4, we introduce formal models for Wikipedia viewed as a temporal resource and for time-based synonyms. In Section 5, we describe our approach to discovering time-based synonyms from Wikipedia. In Section 6, we describe how to use time-based synonyms to improve the retrieval effectiveness. In Section 7, we evaluate our proposed synonym detection and query expansion. Finally, in Section 8, we conclude and outline our future work.

2. RELATED WORK

In recent years, several attempts have been made in using the semi-structured contents of Wikipedia for information retrieval purposes. The ones most relevant to our work are [4, 10, 12, 16, 19, 20]. For a thorough overview of the area of Wikipedia mining, we refer to the survey by Medelyan et al. [11].

In [4], Bunescu and Paşca study how to use Wikipedia for detecting and disambiguating named entities in open domain texts in order to improve search quality. By recognizing entities in the indexed text, and disambiguating between multiple entities sharing the same proper name, the users can access to a wider range of results as today's search engines may easily favor the most common sense of an entity, making it difficult to get a good overview of the available information for a lesser known entity.

An initial approach for synonym detection based on [4] in a non-temporal context was described in [3]. As far as we know, all previous approaches to synonym detection from Wikipedia have been based on redirects only (i.e., [6, 17, 18]) and no temporal aspects are considered.

There is some work that exploits Wikipedia for query expansion. In [10], they proposed to improve the retrieval effectiveness of ad-hoc queries using a local repository of Wikipedia as an external corpus. They analyzed the categorical information in each Wikipedia article, and select terms from top-k articles to expand a query. Then, a second retrieval on the target corpus is performed. Results show that Wikipedia can improve the effectiveness of weak queries while pseudo relevance feedback is unable to improve.

Milne et al. [12] proposed an approach to help users to evolve queries interactively, and automatically expand queries with synonyms using Wikipedia. The experiments show an improvement in recall. The recent work by Xu et al. [19] tackled with a problem of pseudo-relevance feedback that one or more of the top retrieved documents may be non-relevant, which can introduce noise into the feedback process. The proposed approach in [19] classifies queries into 3 categories (entity, ambiguous, and broader queries) based on Wikipedia, and use a different query expansion method for each query category. Their experiments show that Wikipedia based pseudo-relevance feedback improves the retrieval effectiveness, i.e., Mean Average Precision.

To our knowledge, query expansion using synonyms for a temporal search has not been previously described. However, some work related to temporal search exists, including [1, 5, 7, 13, 15], where a user can explicitly specify time as a part of query (called temporal query). Typically, a temporal query is composed of search keywords and temporal criteria, which can be a time point or a time interval. Documents are retrieved by their relevance wrt. the keywords and corresponding temporal criteria.

3. PRELIMINARIES

In this section, we first briefly outline our document and text stream models. Then, we will give a brief overview of Wikipedia articles and the New York Time Annotated corpus.

3.1 Models of Documents and Text Streams

In our context, a document collection contains a number of documents defined as $C = \{d_1, \ldots, d_n\}$. A document can be seen as bag-of-word (an unordered list of terms, or features) with its associated time interval (from it was created until replaced by a new version or deleted): $d_i = \{\{w_1, w_2, w_3, \ldots, w_n\}, [t_i, t_{i+1}]\}$ where $[t_i, t_{i+1}]$ is a time interval of the document, i.e., a time period that d_i exists, and $t_i < t_{i+1}$. $Time(d_i)$ is a function that gives a creation date of the document and must be valid within the time interval, and $Time(d_i) \in [t_i, t_{i+1}]$.

Document collections where their contents appear in a temporal order can be viewed as text streams, e.g. personal emails, news articles and blogs. In such domains, terms in the text streams are temporally dynamic in pattern, e.g., rising sharply in frequency, growing in intensity for a period of time, and then fading away.

3.2 Wikipedia

Wikipedia is a freely available source of knowledge. Each editable article in Wikipedia has associated revisions, i.e., all previous versions of its contents. Each revision (or a version) of an article is also associated with a time period that it was in use before being replaced by the succeeding version. In other words, the time of a revision is a time period when it was a current version.

There are four Wikipedia features that are particularly attractive as a mining source when building a large collection of named entities: article links (internal links in one Wikipedia article to another article), redirect pages (send a reader to another article), disambiguation pages [2] (used by Wikipedia to resolve conflicts between terms having multiple senses by either listing all the senses for which articles exist), and categories (used to group one or more articles together, and every article should preferably be a member of at least one category although this is not enforced).

3.3 New York Time Annotated Corpus

The New York Times Annotated corpus is used in the synonym time improvement task. This collection contains over 1.8 million articles covering a period of January 1987 to June 2007. 1.5 million articles are manually tagged of vocabulary of people, organizations and locations using a controlled vocabulary that is applied consistently across the collections. For instance, if one article mentions "Bill Clinton" and another refers to "President William Jeffer-

[2]Note that the meaning of the term *disambiguation* in Wikipedia context is slightly different from how it is used in computational linguistics.

Table 1: NYT collection statistics of tagged vocabulary

Tagged Vocabulary	#Tagged Documents	#Tagged Documents (%)
People	1.32M	72
Locations	0.6M	32
Organizations	0.6M	32

son Clinton", both articles will be tagged with "CLINTON, BILL". Some statistics of tagged documents are given in Table 1.

4. TEMPORAL MODELS OF WIKIPEDIA

In this section, we will present temporal models of Wikipedia, i.e., synonym snapshots. The models will be later used for detecting synonyms over time. Finally, we will give a formal definition of four different classes of synonyms, and how to classify them using temporal patterns of occurrence as a feature.

4.1 Synonym Snapshots

In our context, a document collection is Wikipedia $\mathcal{W}$ that consists of a set of articles or pages, $\mathcal{P} = \{p_1, \ldots, p_n\}$. There are mainly two types of pages in $\mathcal{P}$: 1) those that describe a named entity, e.g., a concept about people, companies, organizations, countries, etc., and 2) those that do not describe a named entity, e.g., user talk pages, category pages, etc. We call a page in the first type *a named entity page* p_e. For simplicity, we will use the term "entity" and "named entity" interchangeably. An entity e_i is represented by terms constituting the title of an entity page p_e. We define $Entity(p_e)$ as a function that gives the title of an entity page p_e, i.e., $e_i = Entity(p_e)$.

Each page $p_i \in \mathcal{P}$ consists of: 1) terms $\{w_1, \ldots, w_n\}$ where each $w_i \in \mathcal{V}$ and $\mathcal{V}$ is the complete set of terms or a vocabulary in the collection, and 2) a time interval $[t_a, t_b]$, i.e., a time period that p_i exists in the collection: $p_i = \{\{w_1, \ldots, w_n\}, [t_a, t_b]\}$.

We define *TInterval*(x) as a function that gives a time interval associated to x, i.e., a time period of existence $[t_y, t_z]$. We also define *TStart*(x) as a function that gives the starting time point of x, i.e., the smallest time point t_y from the time interval $[t_y, t_z]$ of x, and *TEnd*(x) as a function that gives the ending time point of x, i.e., the largest time point t_z from the time interval $[t_y, t_z]$ of x.

A page p_i is composed of a set of its revisions $\{r_j | r_j \in \mathcal{R}_i\}$. A revision r_j consists of 2 parts: 1) terms $\{w_1, \ldots, w_m\}$, and 2) a time interval $[t_c, t_d)$, or *TInterval*(r_j). Thus, a revision $r_j = \{\{w_1, \ldots, w_m\}, [t_c, t_d)\}$. Note that a time interval of any r_j excludes its last time point, $[t_c, t_d) = [t_c, t_d] - \{t_d\}$.

A revision r_j exists at a different time interval in *TInterval*(p_i), and *TInterval*$(r_j) \subset$ *TInterval*(p_i) and $\cap$*TInterval*$(r_j) = \emptyset$. An intersection of *TInterval*(r_j) of each revision r_j is an empty set because at any time point t in *TInterval*(p_i), only one revision r_j can exist for p_i. Time intervals of two adjacent revisions are defined in term of each other as *TInterval*$(r_j) = [$*TStart*$(r_j),$ *TStart*$(r_{j+1}))$, and *TInterval*$(r_{j+1}) = [$*TEnd*$(r_j),$ *TEnd*$(r_{j+1}))$.

By partitioning $\mathcal{W}$ wrt. a time granularity g, we will have a set of snapshots of Wikipedia $\mathbb{W} = \{W_{t_1}, \ldots, W_{t_z}\}$. In our work, we only use the *1-month* granularity. Hence, if we have the history of Wikipedia for 8 years and $g = month$, the number of snapshots will be $|\mathbb{W}| = 8 * 12 = 96$, i.e., $\mathbb{W} = \{W_{03/2001}, \ldots, W_{03/2009}\}$. Each snapshot W_{t_k} consists of the current revision r_c of every page p_i at time t_k, i.e.,

$W_{t_k} = \{r_c | \forall p_i : r_c \in \mathcal{R}_i \wedge t_k \in \textit{TInterval}(r_c) \wedge \cap \textit{TInterval}(r_c) \neq \emptyset\}$

Because all revisions are current at time t_k, an intersection of their time intervals, $\cap$*TInterval*(r_c), is not an empty set. Figure 1 depicts a snapshot W_{t_k} of Wikipedia and current revisions at time $t = t_k$.

Let $\mathcal{S}$ be a set of synonyms of all entities in $\mathcal{W}$, $\mathcal{S} = \{s_1, \ldots, s_m\}$

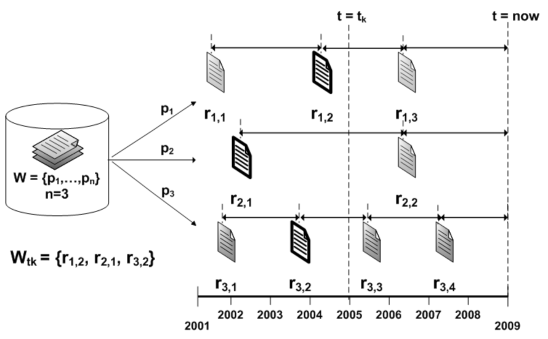

Figure 1: Wikipedia snapshot at time t_k and its current revisions

where each synonym $s_j \in \mathcal{V}$. An entity e_i is composed of a set of synonyms $\{s_1, \ldots, s_u\}$ associated to it. We define $\xi_{i,j}$ as a relationship between an entity e_i and its synonym s_j, that is $\xi_{i,j} = (e_i, s_j)$. Instead of referring to a synonym s_j alone, we have to always refer to an entity-synonym relationship $\xi_{i,j}$, because s_j can be a synonym of one or more entities. Each entity-synonym relationship $\xi_{i,j}$ has an associated time interval $[t_\alpha, t_\beta]$, i.e., a time period that s_j is a synonym of e_i. We can also determine $[t_\alpha, t_\beta]$, t_α and t_β from using functions *TInterval*$(\xi_{i,j})$, *TStart*$(\xi_{i,j})$, and *TEnd*$(\xi_{i,j})$ respectively. We define S_{t_k} as *a synonym snapshot* as a set of entity-synonym relationships at a particular time $t = t_k$. $S_{t_k} = \{\xi_{1,1}, \ldots, \xi_{n,m}\}$ where $t_k \in$ *TInterval*$(\xi_{i,j})$.

In the following subsection, we will explain how to define different classes of synonyms based on occurrence patterns over time.

4.2 Time-based Classes of Synonyms

In this section, we give the definition of time-based classes of synonyms. The intuition behind the synonyms classes is that, synonyms occur differently over time, so they should be employed differently as well. Consequently, we will classify synonyms into different classes based on their occurrence patterns over time.

Let t^w_α be the starting time point and t^w_β be the last time point of the document collection, i.e., Wikipedia. Hence, $t^w_\alpha =$ *TStart*$(\mathcal{W})$ and $t^w_\beta =$ *TEnd*$(\mathcal{W})$. For every entity-synonym relationship $\xi_{i,j}$, let $t^{\xi_{i,j}}_\alpha$ be the first time point we observe $\xi_{i,j}$ and $t^{\xi_{i,j}}_\beta$ be the last time point we observe $\xi_{i,j}$, so $t^{\xi_{i,j}}_\alpha =$ *TStart*$(\xi_{i,j})$ and $t^{\xi_{i,j}}_\beta =$ *TEnd*$(\xi_{i,j})$. Figure 2 depicts occurrence patterns of different synonym classes over time.

Definition 1. *An entity-synonym relationship $\xi_{i,j}$ is classified as **time-independent** (Class A) if all of the following conditions hold:*

(i) $t^{\xi_{i,j}}_\alpha \in [t^w_\alpha, t^w_\alpha + \delta_1]$ *where* $\delta_1 > 0$

(ii) $t^{\xi_{i,j}}_\beta = t^w_\beta$

The idea of Class A is to detect synonyms that exist for a long time interval, as long as that of Wikipedia. These synonyms are robust to change over time and can represent good candidates of synonyms. For example, the synonym "Barack Hussein Obama II" is a time-independent synonym of the entity "Barack Obama". We use δ_1 to relax a condition of starting time because there are not many pages created at the beginning of Wikipedia. For example, δ_1 can be 24 months after Wikipedia was created.

Definition 2. *An entity-synonym relationship $\xi_{i,j}$ is classified as **time-dependent** (Class B) if all of the following conditions hold:*

(i) $t^{\xi_{i,j}}_\alpha, t^{\xi_{i,j}}_\beta \in [t^w_\alpha + \delta_1, t^w_\beta - \delta_2]$ *where* $\delta_2 > 0, t^{\xi_{i,j}}_\alpha > t^{\xi_{i,j}}_\beta$

(ii) $\lambda_1 \leq t_\beta^{\xi_{i,j}} - t_\alpha^{\xi_{i,j}} \leq \lambda_2$ *where* $\lambda_1, \lambda_2 > 0, \lambda_2 > \lambda_1$

The idea of Class B is to detect synonyms that are highly related to time, for example, "Cardinal Joseph Ratzinger" is a synonym of "Pope Benedict XVI" before 2005. We interest in using this synonym class for query expansion to handle the effect of rapidly changing synonyms over time as explained in Section 1. δ_2 indicates that synonyms are no longer in use, and it can be 12 months. λ_1, λ_2 represents minimum, maximum values of a time interval of synonym respectively. For example, λ_1 and λ_2 can be 2 months and 24 months. If a time interval is less than 2 months, it is a noise or junk synonym, and if it is greater than 24 months, it is less specific to time.

In addition to Class A and B, we also observe that there are synonyms that cannot be classified into the two classes above because of their temporal characteristics. Thus, we introduce the fuzzy-membership classes as follows.

Definition 3. *An entity-synonym relationship* $\xi_{i,j}$ *is classified as* ***gaining synonymy*** *(Class C) if all of the following conditions hold:*

(i) $t_\alpha^{\xi_{i,j}} \in [t_\alpha^w + \delta_1, t_\alpha^w + \delta_1 + \epsilon]$ *where* $\epsilon > 0$

(ii) $t_\beta^{\xi_{i,j}} = t_\beta^w$

The idea of Class C is to detect synonyms that exist for a long time interval, *but not* as long as that of Wikipedia. These synonyms can be considered good candidates of synonyms as they are tentative to robust to change over time. However, it is *not* confident to judge if they are time-independent or not. This class of synonyms is actually a special type of Class A that lacks of data in early years. For example, the synonym "Pope" has occurred as a synonym of the entity "Pope Benedict XVI" in 04/2005. Hence, this synonym will be classified to Class C instead of Class A because of its time interval. ϵ is a parameter for the missing data of early years, e.g., ϵ can be 24 months.

Definition 4. *An entity-synonym relationship* $\xi_{i,j}$ *is classified as* ***declining synonymy*** *(Class D) if all of the following conditions hold:*

(i) $t_\alpha^{\xi_{i,j}} \in [t_\alpha^w, t_\alpha^w + \delta_1]$

(ii) $t_\beta^{\xi_{i,j}} \in \left[t_\beta^w - \theta - \delta_2, t_\beta^w - \delta_2\right]$ *where* $\theta > 0$

The idea of Class D is to detect synonyms that are stopped using as synonyms for some time ago, i.e., not in use at the moment. We can consider this class of synonym as *out-of-date* synonyms. For example, for the entity "Bill Clinton", the synonym "President Clinton" is less popular nowadays and it is very rare to be used. Thus, this synonym will belong to Class D. Synonyms in this class can be viewed as a special type of Class B. They are equivalent to synonyms in the past, but their time intervals are not too specific to particular time, i.e., greater than a certain period of time. The period of time is determined by θ that can be 12 months.

5. TIME-BASED SYNONYM DETECTION

In this section, we will present our approach to find time-based entity-synonym relationships. The algorithm is divided into three main steps: 1) named entity recognition and synonym extractions, 2) improving time of synonyms using a model for temporal dynamics of text streams, and 3) synonym classification.

5.1 Named Entity Recognition and Synonym Extraction

First, we partition the Wikipedia collection according to the time granularity $g = month$ in order to obtain a set of Wikipedia snapshots $\mathbb{W} = \{W_{t_1}, \ldots, W_{t_z}\}$.

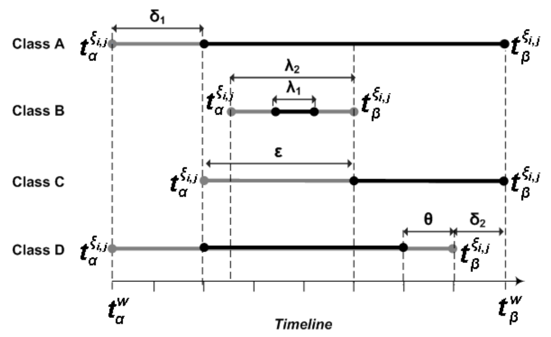

Figure 2: Temporal patterns of time-based classes of synonyms

For each Wikipedia snapshot W_{t_k}, we identify all entities in a snapshot W_{t_k}. A result from this step will be a set of entities E_{t_k} at a particular time t_k. After that, we determine a set of synonyms for each entity $e_i \in E_{t_k}$ in this snapshot W_{t_k}. A result from this process is a set of entity-synonym relations, that is a synonym snapshot $S_{t_k} = \{\xi_{1,1}, \ldots, \xi_{n,m}\}$. We repeat this process for every Wikipedia snapshot W_{t_k} in $\mathbb{W}$. The final result will be a union of all synonym snapshots $\mathbb{S} = \{S_{t_1} \cup \ldots \cup S_{t_z}\}$. $\mathbb{S}$ will be input of the time-based synonym classification step.

Step 1: Recognizing named entities. Given a Wikipedia snapshot W_{t_k}, we have a set of pages existing at time t_k, that is $W_{t_k} = \{p_i | \forall p_i : t_k \in \mathit{TInterval}(p_i)\}$. In this step, we only interest in an entity page p_e. In order to identify an entity page, we use the approach described by Bunescu and Paşca in [4] which is based on the following heuristics:

- If multi-word title with all words capitalized, except prepositions, determiners, conjunctions, relative pronouns or negations, consider it an entity.
- If the title is a single word, with multiple capital letters, consider it an entity.
- If at least 75% of the occurrences of the title in the article text itself are capitalized, consider it an entity.

After identifying an entity page p_e from a snapshot W_{t_k}, we will have a set of entity pages $\mathcal{P}_{e,t_k} = \{p_e | p_e \in W_{t_k}\}$. From this set, we will create a set of entities E_{t_k} at time t_k by simply extracting a title from each entity page $p_e \in \mathcal{P}_{e,t_k}$. A result from this step is a set of entities $E_{t_k} = \{e_1, \ldots, e_n\}$, which will be used in step 2.

Step 2: Extracting synonyms. After identifying a set of entities E_{t_k}, we want to find synonyms for each entity $e_i \in E_{t_k}$. Owing to its richness of semantics structure, it is possible to use article links and redirect pages in Wikipedia for finding synonyms. However, we will not use redirect pages in this paper because it is problematic to define a temporal model of redirect pages. Hence, we will find synonyms by extracting anchor texts from article links. For a page $p_i \in W_{t_k}$, we list all internal links in p_i but only those links that point to an entity page $p_e \in \mathcal{P}_{e,t_k}$ are interesting. In other words, the system extracts as synonyms all anchor texts for the associated entity, and these synonyms are weighted by their frequencies of occurrence. We then obtain a set of entity-synonym relationships. By accumulating a set of entity-synonym relationships from every page $p_i \in W_{t_k}$, we will have a set of entity-synonym relationships at time t_k, i.e., a synonym snapshot $S_{t_k} = \{\xi_{1,1}, \ldots, \xi_{n,m}\}$.

Step 1 and 2 are processed for every snapshot $W_{t_k} \in \mathbb{W}$. Finally, we will obtain a set of entity-synonym relationships from all snapshots $\mathbb{S} = \{S_{t_1}, \ldots, S_{t_z}\}$, and a set of synonyms for all entities $\mathcal{S} = \{s_1, \ldots, s_y\}$. Table 2 depicts examples of entity-synonym relationships and their time periods extracted from Wikipedia. Note

Table 2: Entity-synonym relationships and time periods

Named Entity	Synonym	Time Period
Pope Benedict XVI	**Cardinal Joseph Ratzinger**	**05/2005 - 03/2009**
	Joseph Ratzinger	05/2005 - 03/2009
	Pope Benedict XVI	05/2005 - 03/2009
Barack Obama	Barack Hussein Obama II	02/2007 - 03/2009
	Sen. Barack Obama	07/2007 - 03/2009
	Senator Barack Obama	05/2006 - 03/2009
Hillary Rodham Clinton	Hillary Clinton	08/2003 - 03/2009
	Sen. Hillary Clinton	03/2007 - 03/2009
	Senator Clinton	11/2007 - 03/2009

that, time periods of some relationships in Table 2 are incorrect. For example, the synonym "Cardinal Joseph Ratzinger" of the entity "Pope Benedict XVI" should associates with a time period before 2005. Consequently, in order to improve time periods, the results from this step will be input to the next subsection.

5.2 Improving Time of Entity-synonym Relationships

The time periods of entity-synonym relationships do not always have the desired accuracy. The main reason for this is that the Wikipedia history has a very short timespan of only 8 years. That is, the time periods of synonyms are timestamps of Wikipedia articles in which they appear, not the time extracted from the contents of Wikipedia articles. Consequently, the maximum timespan of synonyms has been limited by the time of Wikipedia. In order to discover the more accurate time, we need to analyze a document corpus with the longer time period, i.e., the New York Time Annotated corpus (NYT).

There are a number of methods for extracting the more accurate time of synonyms. The easiest method is to find the starting time and the ending time, or the first point and the last point in the corpus, at which a synonym is observed with its frequency greater than a threshold. However, the problems with this method are that 1) it cannot deal with sparse/noisy data, or 2) it cannot find multiple, discontinuous time intervals of a synonym.

Alternatively, we can apply the method called "burst detection", proposed in [9] for detecting the time periods of synonyms from the corpus. Bursts are defined as points where a frequency of term increases sharply, and the frequency may oscillate above and below the threshold, resulting in a single long interval of burst or a sequence of shorter ones. Consequently, burst periods can formally represent periods that synonyms are "in use" over time.

The advantage of this method is that it is formally modeled and capable of handling sparse/noisy data. In addition, it can identify multiple, discontinuous time intervals for all terms in the document corpus. Due to the limited space in this paper, readers can refer to [9] for detailed description of the algorithm for burst detection.

We propose to improve the time period of each entity-synonym relationship $\xi_{i,j} \in \mathbb{S}$ by analyzing the NYT corpus (with the longer timespan of 20 years) using the burst detection algorithm. The process of detecting entity-synonym relationships from the NYT corpus is as follows. First, we have to identify a synonym s_j from document streams. Note the difference between an entity-synonym relationship $\xi_{i,j}$ and a synonym s_j, the first one refers to a tuple of synonym s_j and its associated named entity e_i, while the latter one refers to a synonym s_j only.

Second, we have to find a named entity e_i associated to the identified synonym s_j because s_j can be a synonym of more than one named entity. We call this process *synonym disambiguation*. Finally, after we disambiguate synonyms, we will then obtain bursty periods of each entity-synonym relationship $\xi_{i,j}$ that can be represented more accurate time periods of $\xi_{i,j}$.

Table 3: Synonyms and corresponding named entities

#Named Entity	#Synonym
1	2,524,170
2	14,356
3	2,797
4	994
5	442
6	259
7	155
8	94
9	58
10	37

Table 4: Synonyms with different n-grams

N-gram	Synonym
2	Jospeph Ratzinger
3	Senator Barack Obama
5	George III of Great Britain
6	United Nations Commission on Human Rights
8	Society for the Prevention of Cruelty to Animals
13	Queen Elizabeth II of the United Kingdom of Great Britain and Northern Ireland

5.2.1 Identifying and Disambiguating Synonyms using the NYT corpus

To identify a synonym s_j from the text streams of the NYT corpus is not straightforward, because a synonym s_j can be ambiguous (i.e., a synonym may be associated with more than one named entities as Table 3 shows the number of synonyms associated with the different number of named entities). For example, there are more than 19,000 synonyms associating with more than one named entities, while 2.5 million synonyms associate with only one named entities. In order to disambiguate a named entity e_i for a synonym s_j, we can make use of a controlled vocabulary of the NYT corpus described in Section 3.

Recall that input of this step is a set of all synonyms of all entities $\mathcal{S}$ obtained from Subsection 5.1. The algorithm for identifying a synonym s_j from the text streams is given in Algorithm 1 and Algorithm 2. An explanation is as follows. Algorithm 1 finds a synonym s_j from each document d_n where s_j can have the maximum size of n-grams of, or w called the window size of synonym. In this case, a synonym that its size is greater than w is not interesting. Table 4 shows synonyms with different n-grams.

First, read a term s_j with the maximum size w from a document d_n starting at the index pointer *ptr* $= 0$ as in Algorithm 2 (line 7). Check whether s_j is a synonym ($s_j \in \mathcal{S}$), and retrieve all associated named entities for s_j as in Algorithm 2 (line 9). Next, check if s_j has only one associated named entity, then s_j is not ambiguous, as in Algorithm 2 (line 10-11). If s_j is associated with more than one named entities, disambiguate its named entities as in Algorithm 2 (line 13-15). After disambiguating the named entities for s_j, insert an entity-synonym relationship (e_i, s_j) plus the timestamp of d_n, i.e., $Time(d_n)$, in the output set and move the index pointer by the size of s_j, that is *ptr* $=$ (*ptr*$+w$) in Algorithm 1(line 11-12).

If s_j cannot be disambiguated, s_j will be ignored and we continue identifying another synonym, i.e., reading a term with the maximum size w from d_n by increasing the index pointer to the next word *ptr* $=$ (*ptr* $+ 1$) as in Algorithm 1 (line 14). On the contrary, if a term s_j is not a synonym ($s_j \notin \mathcal{S}$), decrease a window size by 1 as in Algorithm 1 (line 20), and consider a prefix string of s_j with a size of $(w - 1)$, or s_{j+1}. If s_{j+1} is not a synonym, repeat the same process until a window size w is equal to 0 as in Algorithm 2 (line 4). This means, if no any prefix substring of s_j

Algorithm 1 *IdentifyEntitySynonymInNYT($\mathcal{D}_\mathcal{N}$)*

```
1: INPUT: $\mathcal{D}_\mathcal{N}$ is a set of documents in the NYT corpus.
2: OUTPUT: A sequence of $\xi_{i,j}$ or $(e_i, s_j)$ and its timestamp.
3: $C \leftarrow \emptyset$    // A set of entity-synonyms relationships and a time point.
4: for each $\{d_n \in \mathcal{D}_\mathcal{N}\}$ do
5:     $len_d \leftarrow |d_n|$    // $len_d$ is the number of words in $d_n$.
6:     $ptr \leftarrow 0$    // ptr is an index pointer in $d_n$, default is 0.
7:     $w \leftarrow c$    // w is the window size of synonym, default is c.
8:     while $ptr \leq len_d$ do
9:         $(e_i, s_j) \leftarrow$ FindSynonym$(d_n, ptr, w)$
10:        if $(e_i, s_j) \neq$ null then
11:            $C \leftarrow C \cup \{(e_i, s_j), Time(d_n)\}$    // Output $(e_i, s_j)$ and
               timestamp of $d_n$
12:            $ptr \leftarrow (ptr +$ CountWords$(s_j))$ // Move ptr by the number
               of words in $s_j$.
13:        else
14:            $ptr \leftarrow (ptr + 1)$    // Move ptr to the next word.
15:        end if
16:    end while
17: end for
18: return $C$
```

Table 5: Tuples of entity-synonym relationships

Timestamp	Entity	Synonym	Frequency
01/1987	President Reagan	Ronald Reagan	54
01/1989	President Reagan	Ronald Reagan	34
10/1990	President Reagan	Ronald Reagan	12
05/2002	Senator Clinton	Hillary Rodham Clinton	121
11/2004	Senator Clinton	Hillary Rodham Clinton	61
01/2005	Senator Clinton	Hillary Rodham Clinton	359

has been recognized as a synonym, continue to read the next term with the maximum size w from the text streams by increasing the index pointer to the next word $ptr = (ptr + 1)$ as in Algorithm 1 (line 14).

After identifying s_j as a synonym, it is necessary to determine whether s_j is ambiguous or not. Note that we retrieve the set of all entities E_j associated with s_j as in Algorithm 2 (line 9). If there is only one entity in E_j, s_j is not ambiguous and that entity will be assigned to s_j as in Algorithm 2 (line 10-11). However, if there are more than one entity, s_j have to be disambiguated by using controlled vocabulary V_n tagged in the document d_n as in Algorithm 2 (line 13).

The algorithm for disambiguating named entities for a synonym is given in Algorithm 3. For each entity $e_k \in E_j$, if e_k is in a set of tagged vocabulary V_n of d_n, add e_k into a list of disambiguated entities E_{tmp} as in Algorithm 3 (line 7-8). Continue for all entities in E_k. If E_{tmp} contains only one entity, s_j is disambiguated. If E_{tmp} has more than one entity, s_j cannot be disambiguated.

The final results will be tuples of disambiguated entity-synonym relationships associated with timestamps of documents where they occur. Table 5 illustrates results from this step of the synonyms "President Reagan" and "Senator Clinton" of the named entities "Ronald Reagan" and "Hillary Rodham Clinton" respectively. Each tuple is composed of an entity-synonym relationship, the timestamp of a document where it occurs, and its frequency. Note that, one entity-synonym relationship can be associated to *different* timestamps. This is equivalent to the statistics of a entity-synonym relationship over time extracted from text streams of documents. The results from this step will be input to the next subsection.

5.2.2 *Improving Time of Synonyms using Burst Detection*

In this step, we will find the correct time of a entity-synonym relationship $\xi_{i,j}$ by using the burst detection algorithm described in

Algorithm 2 *FindSynonym(d_n, ptr, w)*

```
1: INPUT: A document $d_n$, a pointer $ptr$, a size of synonym $w$.
2: OUTPUT: An entity-synonym relationship $(e_i, s_j)$ or $\xi_{i,j}$.
3: $(e_i, s_j) \leftarrow$ null    // Set a tuple result to null.
4: if $w = 0$ then
5:     return $(e_i, s_j)$
6: else
7:     $s_j \leftarrow$ ReadString$(d_n, ptr, w)$    // Read $s_j$ from $d_n$ at index ptr.
8:     if $s_j \in \mathcal{S}$ then
9:         $E_j \leftarrow$ GetAssocEntities$(s_j)$    // All entities associated to $s_j$.
10:        if $|E_j| = 1$ then
11:            $e_i \leftarrow E_j$.firstElement()
12:        else
13:            $e_k \leftarrow$ Disambiguate$(d_n, E_j)$    // Disambiguate $E_j$.
14:            if $e_k \neq$ null then
15:                $e_i \leftarrow e_k$
16:            end if
17:        end if
18:        return $(e_i, s_j)$
19:    else
20:        FindSynonym$(d_n, ptr, (w-1))$  // Find a synonym with a size
           $(w-1)$.
21:    end if
22: end if
```

Algorithm 3 *Disambiguate(d_n, E_j)*

```
1: INPUT: A document $d_n$, and a set of associated entities $E_j$.
2: OUTPUT: A disambiguated entity.
3: $E_{tmp} \leftarrow \emptyset$    // A temporary list of entities.
4: $e_i \leftarrow$ null    // An output entity.
5: $V_n \leftarrow$ GetVocabulary$(d_n)$    // Tagged vocabulary of $d_n$.
6: for each $e_k \in E_j$ do
7:     if $e_k \in V_n$ then
8:         $E_{tmp} \leftarrow E_{tmp} \cup \{e_k\}$
9:     end if
10: end for
11: if $|E_{tmp}| = 1$ then
12:     $e_i \leftarrow E_{tmp}$.firstElement()
13: end if
14: return $e_i$
```

[9]. The algorithm takes the results from the previous step as input, and generates bursty periods of $\xi_{i,j}$ by computing a rate of occurrence from document streams. An output produced in this step is bursty intervals and bursty weight, which are corresponding to periods of occurrence and the intensity of occurrence respectively, as showed in Table 6.

Detected bursty periods are mostly composed of discontinuous intervals because the algorithm depends heavily on a frequency of $\xi_{i,j}$ in the text streams. The disconnect of time intervals prevents us from classifying $\xi_{i,j}$ as time-independent since a time-independent synonym should have a long and continuous time interval. A solution to this problem is to combine two adjacent intervals and interpolate their bursty weight. However, interpolation for $\xi_{i,j}$ will be performed only if a synonym of $\xi_{i,j}$ has no other candidate named entities according to the fact that the relationship of a named entity and its synonym can change over time. A result from this step is a set of entity-synonym relationships, that is $\mathcal{S} = \{\xi_{1,1}, \ldots, \xi_{n,m}\}$ and more accurate time.

5.3 Time-based Synonym Classification

To classify an entity-synonym relationship $\xi_{i,j}$ based on time is straightforward. The starting time point $t_\alpha^{\xi_{i,j}}$ and the ending time point $t_\beta^{\xi_{i,j}}$ of $\xi_{i,j}$ will be used to determine synonym classes as defined in Subsection 4.2. In this work, we are only interested in using time-independent and time-dependent synonyms for query

Table 6: Results from burst-detection algorithm

Synonym	Entity	Burst Weight	Time	
			Start	End
President Reagan	Ronald Reagan	5506.858	01/1987	02/1989
President Ronald	Ronald Reagan	100.401	01/1989	03/1990
President Ronald	Ronald Reagan	67.208	07/1990	02/1993
Senator Clinton	Hillary Rodham Clinton	18.214	01/2001	10/2001
Senator Clinton	Hillary Rodham Clinton	17.732	05/2002	01/2003
Senator Clinton	Hillary Rodham Clinton	172.356	06/2003	11/2004

expansion because synonyms from the other two classes might not be useful in this task. In the next section, we will explain how can we actually make use of time-based synonyms in improving the retrieval effectiveness.

6. QUERY EXPANSION USING TIME-BASED SYNONYMS

In this section, we will describe how to use time-based synonyms (time-independent and time-dependent synonyms) to improve the retrieval effectiveness. The use of synonyms will be divided into two different search scenarios.

The first scenario is to use time-independent class of synonyms in an ordinary search, for example, searching with keywords only (no temporal criteria explicitly provided). The usefulness of time-independent synonyms is that they can be viewed as good candidate synonyms for a named entity. For example, the synonym “Barack Hussein Obama II” is better than “Senator Barack Obama” as a synonym for the named entity “Barack Obama” in this case. Consequently, a query containing named entities can be expanded with their time-independent synonyms before performing a search.

Another case is when performing a temporal search, we must take into account *changes in semantics*. For example, searching documents about “Pope Benedict XVI” written “before 2005”, documents written about “Joseph Alois Ratzinger” should also be considered as relevant because it is a synonym of the named entity “Pope Benedict XVI” at the years “before 2005”. In this case, a time-dependent synonym wrt. temporal criteria can be used to expand a query before searching.

In the rest of this section, we will describe how we actually expand a query with time-based synonyms.

6.1 Query Expansion using Time-independent Synonyms

Before expanding a query and performing an ordinary search, synonyms must be ranked according to their weights. We define a weighting function of time-independent synonyms as a mixture model of a temporal feature and a frequency feature as follows:

$$\mathit{TIDP}(s_j) = \mu \cdot pf(s_j) + (1-\mu) \cdot \overline{tf}(s_j) \qquad (1)$$

where $pf(s_j)$ is a time partition frequency or the number of time partitions (or time snapshots) in which a synonym s_j occurs. $\overline{tf}(s_j)$ is an averaged term frequency of s_j in all time partitions, $\overline{tf}(s_j) = \frac{\sum_i tf(s_j, p_i)}{pf(s_j)}$. μ underlines the importance of a temporal feature and a frequency feature. In our experiments, 0.5 is a good value for μ.

Intuitively, this function measures how popular synonyms are over time. The popularity of synonym over time is measured using two factors. First, synonyms should be robust to change over time as defined in 4.2. Hence, the more partitions synonyms occur, the more robust to time they are. Second, synonyms should have high usages over time. This corresponds to having a high value of averaged frequencies over time.

We intend to use time-independent synonyms in order to improve the effectiveness of an ordinary search, i.e., search without temporal criteria. In this paper, we will perform an ordinary search using Terrier search engine developed by University of Glasgow.

Given a query q, first we have to identify a named entity in query. Note that, we could not rely on state-of-the-art named entity recognition because queries are usually very short (i.e., 2-3 words on average), and lacked of standard form, e.g., all words are lower case. In addition, we need to identify a named entity corresponding to a title of Wikipedia article since our named entities and synonyms are extracted from Wikipedia.

We do this by searching Wikipedia with a query q, and q is a named entity if its search result exactly matches with a Wikipedia page. Besides, a more relax method is to select the top-k related Wikipedia pages instead. Now, we obtain a set of named entities $E_q = \{e_{q,1}, \ldots, e_{q,n}\}$ of q. Subsequently, time-independent synonyms of q are all synonyms corresponding to a named entity $e_{q,i} \in E_q$. Next, we will rank those synonyms by their *TIDP* scores and select only top-k synonyms with highest scores for expansion. Query expansion of time-independent synonyms can be performed in three ways as follows:

1. Add the top-k synonyms to an original query q, and search.
2. Add the top-k synonyms to an original query q, and search with pseudo relevance feedback.
3. Add the top-k synonyms to an original query q plus *TIDP* scores as boosting weight, and search with pseudo relevance feedback.

Boosting weight is a weight of term as defined in Terrier's query language. Note that, if synonyms are duplicated with an original query q, we will remain the original query q unchanged, and add those duplicated synonyms with *TIDP* scores as boosting weight.

6.2 Query Expansion using Time-dependent Synonyms

In order to rank time-dependent synonyms, we first have obtain a set of synonyms from time t_k and weight them differently according to the following weighting function.

$$\mathit{TDP}(s_j, t_k) = tf(s_j, t_k) \qquad (2)$$

where $tf(s_j, t_k)$ is a term frequency of a synonym s_j at time t_k. Note that, a time partition frequency is not counted because synonyms from the same time period should be equal wrt. time. Thus, only a term frequency will be used to measure the importance of synonym.

Time-dependent synonyms will be used for a temporal search, or a search taking into account a temporal dimension, i.e. extending keyword search with a creation or update date of documents. In that way, a search system will retrieve documents according to both text and temporal criteria, e.g., *temporal text-containment search* [13].

Given a temporal query (q, t_k), we will recognize named entities in a query q using the same method as explained in 6.1. After obtaining a set of named entities $E_q = \{e_{q,1}, \ldots, e_{q,n}\}$ of a query q, we will perform two steps of filtering synonyms. First, only synonyms which their time overlaps with time t_k will be processed, that is, $\{s_j | \mathit{Time}(s_j) \cap t_k \neq \emptyset\}$. Second, those synonyms will be ranked by their *TDP* scores and select only top-k synonyms with highest scores for expansion.

Using time-dependent synonyms in a temporal search is straightforward. A set of synonyms will be add into an original temporal query (q, t_k). In the following subsection, we will explain how to automatically generate *temporal queries* that will be later used in temporal search experiments.

7. EXPERIMENTS

In this section, we will evaluate our proposed approaches (extracting and improving time of synonyms, and query expansion using time-based synonyms). Our experimental evaluation is divided into three main parts: 1) extracting entity-synonym relationships from Wikipedia, and improving time of synonyms using the NYT corpus, 2) query expansion using *time-independent synonyms*, and 3) query expansion using *time-dependent* synonyms. In this section, we will describe the setting for each of the main experiments, and then the results.

7.1 Experimental Setting

We will now describe in detail the experimental setting of each of the experiments.

7.1.1 Extracting and Improving Time of Synonyms

To extract synonyms from Wikipedia, we downloaded the latest complete dump of English Wikipedia from the Internet Archive[3]. The dump contains all pages and revisions from 03/2001 to 03/2008 in XML format, and the decompressed size is approximately 2.8 Terabytes. A snapshot was created for every month resulting in 85 snapshots (03/2001, 04/2001, ..., 03/2008). In addition, we obtained 4 more snapshots (05/2008, 07/2008, 10/2008, 03/2009), where 2 of them were downloaded from http://sourceforge.net/projects/wikipedia-miner/files/. So, we have 89 (85+4) snapshots in total.

We used the tool called MWDumper[4] to extract pages from the dump file, and stored the pages and revisions of 89 snapshots in databases using Oracle Berkeley DB version 4.7.25. We then created temporal models of Wikipedia from all of these snapshots.

To improve time of synonyms, we used the burst detection algorithm implemented by the author in [9] and the NYT corpus described in Section 3.3. An advantage of this implementation is that no preprocessing is performed on the documents. Parameter for burst detection algorithm were set as follows: the number of states was 2, the ratio of rate of second state to base state was 2, the ratio of rate of each subsequent state to previous state (for states > 2) was 2, and gamma parameter of the HMM was 1. We use accuracy to measure the performance of our method for improving time of synonyms.

7.1.2 Query Expansion using Time-independent Synonyms

To perform an ordinary search, the experiments were carried out using the Terrier search engine. Terrier provides different retrieval models, such as divergence from randomness models, probabilistic models, and language models. In our experiments, documents were retrieved for a given query by the BM25 probabilistic model with Generic Divergence From Randomness (DFR) weighting. In addition, it provides flexible query language that allows us to specify a boosting weight for a term in query. Given an initial query q_{org}, an expanded query q_{exp} with top-k synonyms $\{s_1, \ldots, s_k\}$ plus *TIDP* scores as boosting weight can be represented in Terrier's query language as follows.

$$q_{exp} = q_{org} \;\; s_1{}^\wedge w_1 \;\; s_2{}^\wedge w_2 \;\; \ldots \;\; s_k{}^\wedge w_k$$

where w_k is a time-independent weight of a synonym s_k, and computed using the function *TIDP*(s_k) defined in Equation 1.

We conducted an ordinary search using the standard Text Retrieval Conference (TREC) collection Robust2004. Robust2004 is the test collection for the TREC Robust Track containing 250 topics (topics 301-450 and topics 601-700). The Robust2004 collection statistics are given in Table 8. The retrieval effectiveness of query expansion using time-independent of synonyms is measured by Mean Average Precision (MAP), R-precision and recall. Recall in our experiments is the fraction of relevant documents Terrier retrieves and all relevant documents for a test query.

[3]http://www.archive.org/details/enwiki-20080103

[4]http://www.mediawiki.org/wiki/Mwdumper

Table 7: Examples of temporal queries and synonyms

Temporal Query		Synonym
Named Entity	Time Period	
American Broadcasting Company	1995-2000	Disney/ABC
Barack Obama	2005-2007	Senator Obama
Eminem	1999-2004	Slim Shady
George H. W. Bush	1988-1992	President George H.W. Bush
George W. Bush	2000-2007	President George W. Bush
Hillary Rodham Clinton	2001-2007	Senator Clinton
Kmart	1987-1987	Kresge
Pope Benedict XVI	1988-2005	Cardinal Ratzinger
Ronald Reagan	1987-1989	Reagan Revolution
Tony Blair	1998-2007	Prime Minister Tony Blair
Virgin Media	1999-2002	Telewest Communications

7.1.3 Query Expansion using Time-dependent Synonyms

To perform a temporal search, we must identify temporal queries used for a search task. We do this in an automatic way by detecting named entities that can represent temporal queries for performing temporal search experiments. Thus, named entities of interesting should have many *time-dependent* synonyms associated to them. To automatically generate temporal queries is composed of two steps as follows.

Given entity-synonym relationships $\mathbb{S} = \{\xi_{1,1}, \ldots, \xi_{n,m}\}$. First, we find temporal query candidates by searching for any named entity e_i which the number of its synonyms is greater than a threshold φ. Nevertheless, in this case, most of synonyms may be *time-independent*, and named entities become less appropriate to represent temporal queries.

Then, we must take into account a *TIDP* of each synonym. The intuition is that the lower *TIDP* weight a synonym has, the better time-dependent it is. So, named entities with an average of *TIDP* weight less than a threshold ϕ probably associate with many *time-dependent* synonyms. This makes them good candidate for temporal queries. In our experiment, the threshold of the number of synonyms φ and a threshold of the average of *TIDP* weight ϕ are 30 and 0.2 respectively.

The temporal searches were conducted by human judgment using 3 users. Some examples of temporal queries are shown in Table 7. Each tuple contains a temporal query (a named entity and time criteria), and its synonym wrt. time criteria. We performed a temporal search by submitting a temporal query to the news archive search engine (`http://www.newslibrary.com`). We compared the results of top-k retrieved documents of each query *without* synonym expansion, and those of the same query *with* synonym expansion. A retrieved document can be either *relevant* or *irrelevant* wrt. temporal criteria. According to the lacking of a standard test set (with all relevant judgments available), we could not evaluate temporal search using recall as we intended. Thus, performance measures are the precision at 10, 20 and 30 documents, or P@10, P@20, and P@30 respectively.

7.2 Experimental Results

First, we will show the results of extracting synonyms, and improving time of synonyms. Then, the results of query expansion us-

Table 8: Robust2004 collection

Source	#Docs	Size	Time Period
Financial Times	210,158	0.56GB	1991-1994
Federal Register	55,630	0.4GB	1994
FBIS	130,471	0.47GB	1996
Los Angeles Times	131,896	0.48GB	1989-1990
Total Collection	528,155	1.9GB	1989-1994, 1996

Table 9: Statistics of entity-synonym relationships extracted from Wikipedia

NER Method	#NE	#NE-Syn.	Max. Syn. per NE	Avg. Syn. per NE
BP-NERW	2,574,319	7,820,412	631	3.0
BPF-NERW	2,574,319	3,199,115	162	1.2
BPC-NERW	473,829	1,503,142	564	3.2
BPCF-NERW	473,829	488,383	148	1.0

Table 10: Accuracy of improving time using the NYT corpus

NER Method	#NE-Syn. Disambiguated		Accuracy (%)
BPF-NERW	393,491	12.3(%)	51
BPCF-NERW	73,257	15.0(%)	73

ing *time-independent* synonyms and the results of query expansion using *time-dependent* synonyms will be presented respectively.

7.2.1 Extracting and Improving Time of Synonyms

To our knowledge, extracting synonyms over time has not been done before. Thus, we could not compare our approach with previous work. However, the statistics obtained from extracting synonyms from Wikipedia are in Table 9. **BP-NERW** is Bunescu and Paşca's named entity recognition of Wikipedia titles described in Section 5.1. **BPF-NERW** is similar to BP-NERW, but we applied *filtering criteria for synonyms*: 1) the number of time intervals is less than 6 months, and 2) the average frequency (the sum of frequencies over all intervals divided by the number of intervals) is less than 2. The filtering aims to remove noise synonyms. **BPC-NERW** is based on BP-NERW, but filtered out named entities that their categories are none of "people", "organization" or "company". **BPCF-NERW** is BPC-NERW with *filtering criteria for synonyms*.

In Table 9, Columns 2-3 are the total number of named entities recognized, and the total number of entity-synonym relationships extracted from Wikipedia, respectively. Column 4 is the maximum number of synonyms per named entity. Column 5 is the average number of synonyms per named entity.

The results from improving time of synonyms using the NYT corpus are in Table 10. Note that, only entity-synonym relationships without noise synonyms are interesting, i.e., recognized by the methods BPF-NERW and BPCF-NERW. In Table 10, Column 2 is the number of entity-synonym relationships that can be identified and assigned time from the NYT corpus using the method in Section 5.2. The percentage of the number of entity-synonym relationships identified and assigned time is shown in Column 3.

In order to evaluate the accuracy of the method for improving time of entity-synonym relationships, we randomly selected 500 entity-synonym relationships and manually assessed the accuracy of time periods assigned to those entity-synonym relationships. The accuracy of the method for improving time of entity-synonym relationships is shown in Column 4. The accuracy of the method for improving time of entity-synonym relationships in a case of BPCF-NERW is better than that of BPF-NERW because named entities recognized by BPF-NERW is too generic, and it is rare to gain high frequencies in the NYT corpus.

7.2.2 Query Expansion using Time-independent Synonyms

The baseline of our experiments is the probabilistic model (**PM**) without query expansion. In addition, we also consider two classical expansion models: reweighing an original query (**RQ**) and pseudo relevance feedback using Rocchio algorithm (**PRF**). Our three proposed expansion methods are: 1) add the top-k synonyms to an original query before search (**SQE**), 2) add the top-k synonyms to an original query and search with pseudo relevance feedback (**SQE-PRF**), and 3) add the top-k synonyms to an original query plus their *TIDP* scores as boosting weight, and search with pseudo relevance feedback (**SWQE-PRF**). Pseudo relevance feedback was performed by selecting 40 terms from top-10 retrieved documents, and those expansion terms were weighted by DFR term weighting model, i.e., Bose-Einstein 1.

Test queries were selected from the Robust2004 test set using named entities in a query described in Section 6.1. Note the difference between Bunescu and Paşca's named entity recognition for Wikipedia page (**BP-NERW**), and named entity recognition in a query (**NERQ**). The first method recognizes whether a Wikipedia document is a named entity or not, and it needs to analyze the content of the Wikipedia document. For the second method, we have only a set of short queries (without a document) and we need to identify named entities in those queries. Recall that there are two methods for recognizing named entities in queries described in Section 6.1: 1) exactly matched Wikipedia page (**MW-NERQ**), and 2) exactly matched Wikipedia page and top-k related Wikipedia pages (**MRW-NERQ**). We used $k = 2$ in our experiments because k greater than 2 can introduce noise to the NERQ process.

The number of queries from the Robust2004 test set recognized using two methods are shown in Table 12. There are total 250 queries from Robust2004. MW-NERQ can recognize 42 named entity queries while MRW-NERQ can recognize 149 named entity queries. Note that, 42 and 149 queries are the number of queries found as Wikipedia article, and recognized as named entities. For example, there are actually 58 queries from Robust2004 found as Wikipedia article, but only 42 are *named entity* queries.

Named-entity queries recognized using two NER methods are shown in Table 11. Each row represents different retrieval results of each retrieval method, and two main column represents two different methods for NERQ. Different retrieval results are composed of Mean Average Precision (MAP), R-precision and recall. As seen in Table 11, our proposed query expansion methods SQE-PRF and SWQE-PRF performs better than the baselines PM, RQ and PRF in both MAP and recall for MW-NERQ. However, there is only SWQE-PRF outperforming the baselines in R-precision. Also note that, SQE-PRF has better recall than SWQE-PRF, while the opposite seems to hold for precision. In the case of MRW-NERQ, our proposed query expansion methods have really worse performance than in the case of MW-NERQ due to the accuracy of the recognition method.

7.2.3 Query Expansion using Time-dependent Synonyms

The baseline of our experiments is to search using a temporal query (**TQ**), i.e., a keyword w_q and time t_q. Our propose method is to expand an original query with synonyms wrt. time t_q and search (**TSQ**). Experimental results of P@10, P@20 and P@30 of **20** of temporal query topics are shown in Table 13. The results show that our query expansion using time-dependent synonyms TSQ performed significantly better than temporal searches without expan-

Table 11: Performance comparisons using MAP, R-precision, and recall for named entity queries, * indicates statistically improvement over the baselines using t-test with significant at $p < 0.05$

Method	MW-NERQ			MRW-NERQ		
	MAP	R-precision	Recall	MAP	R-precision	Recall
PM	0.2889	0.3309	0.6185	0.2455	0.2904	0.5629
RQ	0.2951	0.3266	0.6294	0.2531	0.2912	0.5749
PRF	0.3469	0.3711	0.6944	**0.3002**	**0.3227**	**0.6761**
SQE	0.3046	0.3360	0.6574	0.2123	0.2499	0.5385
SWQE	0.3054	0.3399	0.6475	0.2399	0.2820	0.5735
SQE-PRF	0.3608*	0.3652	**0.7405***	0.2507	0.2665	0.5932
SWQE-PRF	**0.3653***	**0.3861***	0.7388*	0.2885	0.3080	0.6504

Table 12: Number of queries using two different NER

Type	MW-NERQ	MRW-NERQ
Named entity	42	149
Not named entity	208	101
Total	250	250

Table 13: Performance comparisons using P@10, P@20 and P@30 for temporal queries * indicates statistically improvement over the baseline using t-test with significant at $p < 0.05$

Method	P@10	P@20	P@30
TQ	0.1000	0.0500	0.0333
TSQ	**0.5200***	**0.3800***	**0.2800***

sion TQ. Our observation is that TQ retrieved zero to a few relevant documents (less than 10) for most of temporal queries, while TSQ could retrieve more relevant documents as a result of expanding temporal queries with time-dependent synonyms.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have described how to use a Wikipedia to discover time-dependent and time-independent synonym. These classified synonyms can be employed in a number of application areas, and in this paper we have described how to perform query expansion using the time-based synonyms. The usefulness of this approach has been demonstrated through an extensive evaluation, which have showed significant increase in retrieval effectiveness.

Future work include combining time-dependent synonyms and temporal language models in order to provide temporal search using named entity query expansion without having to provide explicitly the time in the query. We will also integrate our approach for time-dependent synonym discovery with information extraction techniques that can find additional information in Wikipedia (for example names of presidents at particular points in time). Finally, we also intend to use the detected relationships in order to improve performance of temporal text-clustering.

9. REFERENCES

[1] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. Fluxcapacitor: efficient time-travel text search. In *Proceedings of the 33rd VLDB*, 2007.

[2] K. Berberich, S. J. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *Proceedings of SIGIR'2007*, 2007.

[3] C. Bøhn and K. Nørvåg. Extracting named entities and synonyms from Wikipedia. In *Proceedings of AINA'2010*, 2010.

[4] R. C. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL'2006*, 2006.

[5] D. Efendioglu, C. Faschetti, and T. Parr. Chronica: a temporal web search engine. In *Proceedings of the 6th ICWE*, 2006.

[6] J. Hu, L. Fang, Y. Cao, H.-J. Zeng, H. Li, Q. Yang, and Z. Chen. Enhancing text clustering by leveraging Wikipedia semantics. In *Proceedings of SIGIR'2008*, 2008.

[7] A. Jatowt, Y. Kawai, and K. Tanaka. Temporal ranking of search engine results. In *Proceedings of WISE*, 2005.

[8] N. Kanhabua and K. Nørvåg. Improving temporal language models for determining time of non-timestamped documents. In *Proceedings of ECDL'2008*, 2008.

[9] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of SIGKDD'02*, 2002.

[10] Y. Li, W. P. R. Luk, K. S. E. Ho, and F. L. K. Chung. Improving weak ad-hoc queries using Wikipedia as external corpus. In *Proceedings of SIGIR'2007*, 2007.

[11] O. Medelyan, D. N. Milne, C. Legg, and I. H. Witten. Mining meaning from Wikipedia. *Int. J. Hum.-Comput. Stud.*, 67(9):716–754, 2009.

[12] D. N. Milne, I. H. Witten, and D. M. Nichols. A knowledge-based search engine powered by Wikipedia. In *Proceedings of CIKM'2007*, 2007.

[13] K. Nørvåg. Supporting temporal text-containment queries in temporal document databases. *Journal of Data & Knowledge Engineering*, 49(1):105–125, 2004.

[14] M. Sanderson. Ambiguous queries: test collections need more sense. In *Proceedings of SIGIR'2008*, 2008.

[15] N. Sato, M. Uehara, and Y. Sakai. Temporal ranking for fresh information retrieval. In *Proceedings of the 6th IRAL*, 2003.

[16] R. Schenkel, F. M. Suchanek, and G. Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. In *Proceedings of BTW'2007*, 2007.

[17] P. Wang, J. Hu, H.-J. Zeng, L. Chen, and Z. Chen. Improving text classification by using encyclopedia knowledge. In *Proceedings of ICDM'2007*, 2007.

[18] F. Wu and D. S. Weld. Autonomously semantifying Wikipedia. In *Proceedings of CIKM'2007*, 2007.

[19] Y. Xu, G. J. Jones, and B. Wang. Query dependent pseudo-relevance feedback based on Wikipedia. In *Proceedings of SIGIR'2009*, 2009.

[20] T. Zesch, I. Gurevych, and M. Mühlhäuser. Analyzing and accessing Wikipedia as a lexical semantic resource. In *Proceedings of Biannual Conference of the Society for Computational Linguistics and Language Technology*, 2007.