IEICE TRANS. INF. & SYST., VOL.E94–D, NO.3 MARCH 2011

PAPER Special Section on Data Engineering

# Query Expansion and Text Mining for ChronoSeeker ——Search Engine for Future/Past Events*——

**Hideki KAWAI**[†,††a)], **Adam JATOWT**[††], **Katsumi TANAKA**[††], *Nonmembers*, **Kazuo KUNIEDA**[†], *and* **Keiji YAMADA**[†], *Members*

**SUMMARY** This paper introduces a future and past search engine, ChronoSeeker, which can help users to develop long-term strategies for their organizations. To provide on-demand searches, we tackled two technical issues: (1) organizing efficient event searches and (2) filtering out noises from search results. Our system employed query expansion with typical expressions related to event information such as year expressions, temporal modifiers, and context terms for efficient event searches. We utilized a machine-learning technique of filtering noise to classify candidates into information or non-event information, using heuristic features and lexical patterns derived from a text-mining approach. Our experiment revealed that filtering achieved an 85% F-measure, and that query expansion could collect dozens more events than those without expansion.

***key words:*** *futurology, strategic foresight, futuristics, scenario planning, vertical search, text mining, information retrieval*

## 1. Introduction

Today's social and economic situations are becoming more complex. Our world is facing the worst financial crisis in a century, and the degree of uncertainty about the future is increasing. Business executives and policy makers need long-term strategic thinking based on future predictions concerning changing trends in society, the economy, science, technology and people's sense of values.

The most important point in future predictions for strategic thinking is not just making good guesses but anticipating every possible scenario including both optimistic and pessimistic views. This is because when something happens, people can cope quickly with problems by choosing the most suitable scenario. It is also important to consider the great variety of opinions regarding the future from many points of view to create more reliable future scenarios.

Comprehensive and thorough investigations are needed to prepare us for various future scenarios. For instance, when considering "the future of Japan," one needs to investigate not only domestic trends but also global trends in economy, society, and scientific technologies. Likewise when considering "the future of education," one needs to investigate advances in e-Learning technology and changes in skill requirements for emerging jobs. Histories are also important for future predictions. For example, a similar situation in the past can be a good hint for predicting what is likely to happen in the future. This is why, in this work, we focused not only on the retrieval of future events but also on the acquisition of information on past events. As evidenced in the experiments, the approach to extract information on past events differs to a certain extent from the one that aims at extractiong information on future events.

There are already some existing databases on future predictions. Sigma Scan [**] contains future information surveying more than 2,000 document sources and interviews with 300 leading thinkers. FutureTimeline [***] contains more than 13,000 future predictions manually collected from news articles, research papers and government reports. The weakness of these manually maintained databases is that it is difficult to always keep their information up-to-date.

Since the Web reflects societal trends, ideas and expectations, there is a great deal of information devoted to past or future events. Examples of past events are the history of the world, companies, and people. Examples of future events are predictions, plans, schedules, and speculations.

Our research goal was to provide an on-demand Japanese search engine for chronological events (CEs) by effectively utilizing an up-to-date index of Web-search application program interfaces (APIs). We defined a CE as a sentence that describes future or past events with an explicit year, $y$. Examples of CEs are future predictions such as "Usage of thin-film solar cells could be up to 30% in 2015", and past events such as "The basic principle of solar cells was invented by a French physicist in 1839." Note that our method was designed for the Japanese language; however, it can be adapted to other languages. Note also that future/past dates can have different granularities, such as months, days and decades. We only focused on year expressions for the sake of simplicity in this paper.

It is difficult for users to manually extract the whole range of CEs related to a given query for two main reasons. First, a single query (e.g., the name of a given object) may not return enough information on related CEs. Thus a

Manuscript received June 7, 2010.
Manuscript revised October 5, 2010.

[†]The authors are with NEC C&C Innovation Research Laboratories, Ikoma-shi, 630–0101 Japan.
[††]The authors are with the Department of Social Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.

*This paper is based on "ChronoSeeker: Search Engine for Future and Past Events," by H. Kawai, A. Jatowt, K. Tanaka, K. Kunieda and K. Yamada, which appeared in the Proceedings of the 4th International Conference on Ubiquitous Information Management and Communication (ICUIMC), Suwon, Korea, January 2010. © 2010 ACM.

a) E-mail: h-kawai@ab.jp.nec.com
DOI: 10.1587/transinf.E94.D.552

**http://www.sigmascan.org/
***http://seikatsusoken.jp/futuretimeline/

combination of specially crafted queries is needed to obtain the whole range of related events. Second, the returned results often contain noisy information, which is problematic for users to manually investigate. Therefore we prescribe a framework in this paper for automatically acquiring temporal information related to user queries, which is aimed at reaching maximum precision and recall. We will address the two technical issues of:

- How the system can collect a large number of CEs related to a user query, $q$, within a limited response time and
- How the system can filter out noisy sentences that are irrelevant to either query $q$ or year $y$.

Our proposed system, ChronoSeeker, uses query expansion and search strategies to collect as many CEs as possible within an acceptable response time. We also applied a machine learning technique to eliminate noise. Finally, the search result for CEs can be visualized not only as a document list in traditional search engines but also as events on a timeline.

The four main contributions of this paper are:

- We propose three types of terms for query expansion to efficiently collect CEs related to search queries,
- We compare statistical and dynamic search strategies with different combinations of query expansion words,
- We find that different query expansion and search strategies should be used for future and past searches, and
- We propose feature selection methods for a machine learning technique to remove noise from search results.

The remainder of this paper is structured as follows. The next section explains the main components of our proposed system, ChronoSeeker. Section 3 describes experimental settings and results, and presents a discussion. Section 4 introduces some related works on future/past event extraction. Section 5 is the conclusion.

## 2. Chronological Event Searches

The simplest method of collecting CEs from the Web is by first issuing user query $q$ to a Web-search API, and extracting year expression $y$ from the retrieved Web pages. However, this method has three main drawbacks:

1. A Web page can contain many date expressions, but only a few of them are actually relevant to query $q$. Kimura et al. estimated that a web page matching a person's name contains 10.5 date expressions on average, but only 13% (166/1308) of these are relevant to the name [10]. This indicates that a web page only contains one or two date expressions relevant to a query $q$.
2. We cannot selectively search for future or past events. Usually, a search result can contain both future and past events about a query $q$, but only few future events would appear in many cases.

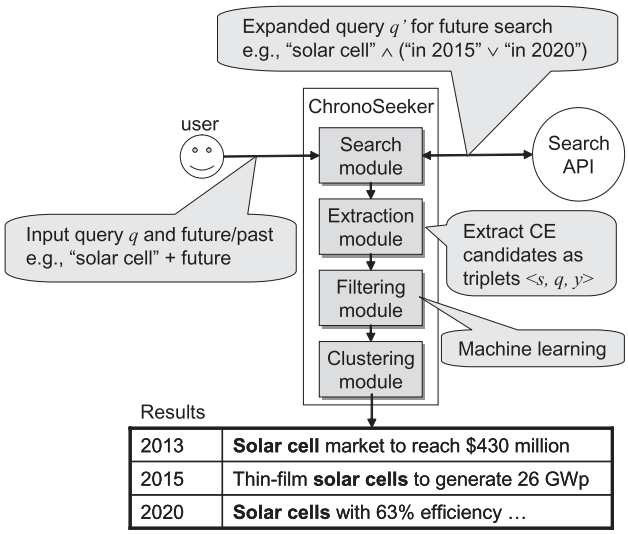


**Fig. 1** Overview of ChronoSeeker.

3. Crawling through and analyzing the whole body of all retrieved pages requires huge network resources. It would take too long if the system fetched entire pages one-by-one to extract year expressions. Even if these processes were done in parallel, downloading hundreds of web pages for one query would not be not feasible.

Our solutions to these problems are as follows.

- We assumed that a snippet in a search result would be a window that contained the most relevant part of the Web page to the search query.
- We expanded user query $q$ to query $q'$ so that the most relevant year expression $y$ to $q$ would appear in the snippet. This method of query expansion could solve the first and second problems.
- We only used snippets to extract year expression $y$. In this way, we could solve the third problem.

Note, however, that many future predictions may not have any exact dates associated whith them as they may simply be unknown or still uncertain. Such information may not be detected with our proposed approach. This also applies to past events. These implicit CEs should be addressed in future work.

Figure 1 has the system overview for ChronoSeeker. It consists of four modules: search, extraction, filtering and clustering. It can also output the extracted CEs for external visualization tools, which can present the CEs as a timeline. ChronoSeeker utilizes a Web-search API as a huge corpus. As a result, we adopted the Boolean-based full-text search model commonly used in many search APIs, where a query, $q$, for retrieving documents that contain both terms $t_1$ and $t_2$ can be represented by a Boolean expression, '$t_1$ AND $t_2$'. For notational convenience, we used the symbol '∧' to denote Boolean operator 'AND' and '∨' to denote 'OR'. We also used double quotation marks to denote phrase searches that only match documents that contain a specified phrase, such as a "solar cell."

First, when users issue a query, $q$, the search module expands query $q$ to $q'$ such that a search API can return snippets that contain both query $q$ and year expression $y$. This query expansion can differ depending on the target period of the search, i.e., future or past. In Fig. 1, query $q$ = "solar cell" for the future search is expanded to $q'$ = "solar cell" ∧ ("in 2015" ∨ "in 2020"). Formally, user query $q$ would be expanded to $q'$ as:

$$q' = q \wedge \left( \bigvee_{i=1}^{n} k_i \right), \tag{1}$$

where $K = \{k_1, k_2, \ldots, k_n\}$ is a set of query expansion terms. Note that $k_i$ consisting of multiple terms such as "in 2010" can be added as a phrase to the query expansion.

Second, the extraction module extracts candidate sentences of CEs that contain year expression $y$. The filtering module determines whether the candidates are actually CEs or not. We applied a machine-learning technique to the filtering process using some features such as characteristic terms in the CE candidates. More details on features will be discussed later.

Next, the clustering module groups CEs that describe the same event based on the terms of the CEs. At this point, the system gives a higher attention score to the bigger cluster. Finally, the system outputs the list of clustered CEs ranked by year and attention level. Users can obtain the original web pages for a certain CE by clicking links. We just applied a simple k-means clustering method and used n-gram models as the feature vector for CEs. The details of clustering are out of focus in this paper.

The following subsections give more details on query expansion and search strategies in the search module, and explain a machine-learning technique and feature vectors in the filtering module.

### 2.1 Query Expansion

As previously mentioned, the purpose of query expansions is to obtain many CE candidates as sentences that contain both user query $q$ and year expression $y$. To achieve this goal, the terms for query expansion should be characteristic expressions in CEs. We propose three types of terms for query expansion, (1) Year Expressions, (2) Temporal Modifiers and (3) Context Terms.

*Year expressions* (YEs) describe the year when a CE happened. There are two kinds of year expressions, absolute and relative. Examples of absolute-year expressions are "1974 年 (1974)" and "2015 年 (2015)". Note that in Japanese, we describe the year as a number followed by a unit "年 (year)". Examples of relative year expressions are "3 年後 (three years later)", "来年 (next year)", "20 年後 (twenty years ago)" and "去年 (last year)". In this research, we only targeted absolute-year expressions to avoid accessing a whole Web page to convert a relative year into an absolute one.

*Temporal modifiers* (TMs) consist of particles and nouns that follow year expressions, and they specify a point or a range of time when a CE happened. Examples of temporal modifiers are "年までに (by the year)", "年に (in the year)", and "年頃に (around the year)". Year expressions can match CEs but also many time stamps of blogs and BBSs, while temporal modifiers usually appear in plain text sentences.

*Context terms* (CTs) are terms that often appear with year expressions together on the same web pages. Examples of context terms for past events are "誕生 (birth)", "起源 (originate)" and "由来 (derive)". Examples for future events are "目標 (goal)", "達成 (achieve)" and "実用化 (practical use)". If a context term appears around query $q$ in a sentence, it is highly possible that the year expression related to the CE can also be found in the same sentence.

There are several concerns about query expansion. As seen in the above examples, query-expansion terms should be changed depending on future or past searches. Also, we need to carefully select only important terms for two main reasons. The first is that a query is usually limited to around 30 words [†]. The second is that if a Web page contains too many terms in the expanded query, the snippets will be too fragmented to extract CEs as sentences.

Another concern is that we cannot use year expressions for past searches because there are too many candidates for past year expressions, and we cannot select related years for the past CEs in advance. However, year expressions for future search can work well because most future predictions treat events emerging within the next decade. Thus, we can expand a query by adding year expressions for ten years from 2010 to 2020.

The combination of different types of query expansion terms is also an interesting topic. Using two different types of query expansion terms $K = \{k_1, k_2, \ldots, k_n\}$ and $K' = \{k'_1, k'_2, \ldots, k'_m\}$, a query $q$ can be expanded to $q'$ as:

$$q' = q \wedge \left( \bigvee_{i=1}^{n} k_i \right) \wedge \left( \bigvee_{j=1}^{m} k'_j \right). \tag{2}$$

We will discuss how can we maximize the number of extracted CEs by combining query-expansion terms based on our experiments.

### 2.2 Search Strategy

A basic approach to develop a long-term strategy is to analyze a wide range of factors of the external macro-environment. Many related keywords, especially new words and concepts can be found during this process because changes and trends for the future to be described as new words. For example, analysts who analyze the future of "electric cars" will find a keyword such as "hydrogen station" as an infrastructure for fuel cell electric vehicles. Then they need to find additional information about the word. So we designed ChronoSeeker as an on-demand search engine

[†]For example, Google limits queries to 32 words.

which allows users to conduct interactive searches finding new keywords as queries. So the "Eight Second Rule" has been used as a standard response time for websites [20], but a recent study has indicated that responding within five seconds is more advantageous [16].

The search strategy in our system should be limited because it needs to access an external-search API and extract CEs from the search results within five to eight seconds. The average response time in the search API measured by our preliminary experiment was around one to two seconds. We also limited the number of processes accessing a server to only one at the same time to avoid placing a heavy load on the server. Due to these constraints, the number of access attempts from the search module of ChronoSeeker to a search API should be limited to no more than twice per query.

Usually, the number of search results is limited. Consequently, we need to call search APIs several times changing the beginning rank, $r$, of the search results to obtain more search results. As previously mentioned, we only can call a search API twice. Even though we imposed this hard constraint, there is still room for applying two different search strategies in the search module, i.e., static and dynamic searches. The search module in a static search issues the same expanded query, $q'$, to obtain the second list of search results. While the search module in a dynamic search expands queries differently when it obtains the second search results. More formally, we can describe a set of snippets returned by search API as $S(q', r, \alpha)$, where $\alpha$ is the number of results returned by one search API call. For a static search, the system obtains $\alpha$ snippets from the top of the search result for $q'$ on the first API call, and obtains $\alpha$ snippets from $\alpha+1$-th result for the same expanded query $q'$ on the second API call. For a dynamic search, the system obtains $\alpha$ snippets from the top of the search results for $q'$ on the first API call, and obtains $\alpha$ snippets from the top of the results for differently expanded query $q''$ on the second API call. Let $S_{sta}$ and $S_{dyn}$ denote sets of snippets collected by static and dynamic searches. These search strategies can be described as:

$$S_{sta}(q') = S(q', 1, \alpha) \cup S(q', \alpha + 1, \alpha), \tag{3}$$

$$S_{dyn}(q', q'') = S(q', 1, \alpha) \cup S(q'', 1, \alpha). \tag{4}$$

We can decide which strategy is best under which conditions based on our experiments. We will also discuss how we can optimize the search strategy when some of these constraints are relaxed with the development of a network technology based on the experimental results.

### 2.3 Chronological Event Filtering with SVM

ChronoSeeker's extraction module extracts year expressions $y$ from a sentence containing user query $q$. However, subsequent processing is required since some noise can occur. Suppose that a sentence, "Brazil and Germany have signed a cooperation agreement for the 2014 World Cup and 2016 Olympic and Paralympics Games to be held in Brazil", is retrieved by inputting the query "Olympic". The extraction module can find two year expressions of "2014" and "2016" from the sentence, but only "2016" is relevant to the query, "Olympic." Therefore, ChronoSeeker's filtering module needs to remove "2014" as noise.

Formally, let $s$ denote a sentence that contains query $q$ and year expressions $Y = \{y_1, y_2, \ldots, y_k\}$. CE candidates derived from sentence $s$ can be denoted by a set of triplets $CE_{cand} = \{< s, q, y_1 >, < s, q, y_2 >, \ldots, < s, q, y_k >\}$. The problem with filtering CE candidates can be defined as attaching label $l$ to every triplet. If the $i$-th year expression $y_i$ is relevant to query $q$ in sentence $s$, we attach label $l_i = +1$ to the triplet, $< s, q, y_i >$. However, we attach $l_i = -1$ to it if $y_i$ is irrelevant to $q$ in $s$. This problem can be easily translated into a classification problem. Thus, we can apply Support Vector Machines (SVMs) [17] as one of the machine-learning techniques for this task. We will now give details on the algorithm for training SVMs for filtering CE candidates. Note that $s$ can not be a full sentence but a partitioned one because only contexts around query terms appear in snippets.

Given a training set of pairs $T = \{(\boldsymbol{x_1}, l_1), \ldots, (\boldsymbol{x_n}, l_n)\}$, $\boldsymbol{x_i} \in \mathbb{R}^d$, $l_i \in \{-1, +1\}$, where $\boldsymbol{x_i}$ is a $d$-dimensional feature vector for a CE candidate $< s, q, y_i >$. The SVM algorithm finds a hyperplane that optimally splits the training set. The optimal hyperplane can be distinguished by using the maximum margin of separation between all training points and the hyperplane. The hyperplane can be described as a decision function like:

$$f(\boldsymbol{x}) = sgn((\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})) + \rho), \tag{5}$$

where $\boldsymbol{w}$ is the weight vector, $\rho$ is a scalar that represents the margin of the hyperplane, $\boldsymbol{\Phi}(\boldsymbol{x})$ is a function that transforms the feature vector into a higher dimensional feature space, and $sgn$ is the sign function. To obtain the weight vector $\boldsymbol{w}$ that leads to the largest $\rho$ of the optimal hyperplane, we solve the following primal optimization problem:

$$minimize: \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i^n \xi_i \tag{6}$$

$$subject\ to: \ l_i(\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x_i}) + \rho) \geq 1 - \xi_i, \ \xi_i \geq 0, \tag{7}$$

where $\xi_i$ is a slack variable and parameter $C$ is a regularization term, which provides a way of controlling overfitting.

### 2.4 Generation of Heuristic Features for SVM

Because we consider the extraction of CEs to be a classification problem, we need to decide how to represent the triplet of CE candidate $< s, q, y_i >$ in terms of a feature vector. The key point is finding features that will help determin whether query $q$ and year expression $y_i$ are relevant in sentence $s$ or not. We employ five features in this paper to represent sentence $s$ as follows:

**Uni-grams (UGs)**

We used uni-grams (only nouns and verbs) as the most

basic features from CE candidates. We also used unigrams extracted from the title of a web page containing the CE candidates. These features are represented as the N-dimensional binary vector $(ug_1, \ldots, ug_N)$, such that $ug_i = 1$ if the sentence or the title of the page contains uni-gram $UG_i$, and $ug_i = 0$ if not, where $N$ is the number of uni-grams appearing in all the CE candidates.

**Temporal Modifiers (TMs)**

Temporal modifiers for query expansion are also used for the feature vectors. This feature is represented as the T-dimensional binary vector, $(tm_1, \ldots, tm_T)$, such that $tm_i = 1$ if the CE candidate contains the temporal modifier, $TM_i$, and $tm_i = 0$ if not, where $T$ is the number of temporal modifiers for query expansion.

**Context Terms (CTs)**

Context terms for query expansion are also used for the feature vectors. This feature is represented as the Z-dimensional binary vector, $(ct_1, \ldots, ct_Z)$ such that $ct_i = 1$ if the CE candidate contains the context term, $CT_i$, and $ct_i = 0$ if not, where $Z$ is the number of context terms for query expansion.

**Same Window (SWs)**

The distance between query $q$ and year expression $y$ in the CE candidate sentence $s$ can also be an important feature for filtering. Thus, we use the $\omega$-term window to represent closeness between $q$ and $y$. This feature can be represented as a binary scalar, $sw$, such that $sw = 1$ if both $q$ and $y$ appear within the window, and $sw = 0$ if not.

**Different Years (DYs)**

If a CE candidate contains many year expressions, the chance that $q$ and $y$ are relevant is lower. This feature can be represented as a binary scalar $dy$, such that $dy = 1$ if different year expression $y'$ appears between $q$ and $y$, and $dy = 0$ if not.

For machine learning, a CE candidate $< s, q, y >$ can be represented as an input vector x appending $q$ and $y$ to the feature vector derived from $s$. For example, when we use UG as the feature vector for $s$, the input vector for machine learning can be represented as $\mathrm{x} = (q, y, ug_1, \ldots, ug_N)$. We also tried combinations of the above features. For notational convenience, we use the symbol '+' to denote an append operation of two feature vectors. For example, UG+TM denotes that we used a feature vector of sentence $s$ appending both vectors of uni-grams and temporal modifiers $(ug_1, \ldots, ug_N, tm_1, \ldots, tm_T)$. In the following experiment, we tried five different combinations of feature vectors: UG, UG+TM, UG+TM+CT, UG+TM+CT+SW and UG+TM+CT+SW+DY.

### 2.5 Text Mining for Feature Generation

We also propose a "Pairwise Pattern (PP)" extraction based on a text-mining approach to generate feature vectors for SVM. The main advantage of our method is that we can generate lexico syntactic patterns from a small set of sentences with a light-weight method. The key point is that we can extract a sequence pattern consisting of common terms between a pair of sentences, and the order of the common terms in the pattern of the original sentence is retained.

**Procedure** GetPairwisePattern

Input: $ce_1$, $ce_2$

1: $W_1 \leftarrow$ parse($ce_1$); $W_2 \leftarrow$ parse($ce_2$)
2: *SubPatterns* $\leftarrow \phi$
3: *pat* $\leftarrow$ ""; *len* $\leftarrow 0$
4: **for** each $w$ in $W_1$ **do**
5: **if** $w \subseteq W_2$ **then**
6: *pat* $\leftarrow$ *pat* + $w$; *len*++
7: **else if** *len* $> \tau$ **then**
8: *add pat to SubPatterns*
9: *pat* $\leftarrow$ ""; *len* $\leftarrow 0$
10: **else**
11: *pat* $\leftarrow$ ""; *len* $\leftarrow 0$
12: **endif**
13: **end for**
14: *PairwisePattern* $\leftarrow$ join *SubPatterns* to a string with "*".
15: **return** *PairwisePattern*

**Fig. 2** Algorithm to generate pairwise pattern.

To illustrate these basic concept, let us assume the following two sentences:

- $s_1$: "With more than 333 million adults expected to have diabetes by 2025."
- $s_2$: "Air passenger numbers expected to more than double by 2030."

As the common words in $s_1$ and $s_2$ are {more, than, expected, to, by}, the sequence pattern $p_{12}$ of the common terms in $s_1$ is "more than * expected to * by", and the sequence pattern, $p_{21}$, appearing in $s_2$ is "expected to more than * by", where the asterisk "*" denotes a wild card that can match any sequence of terms. Please note that we can obtain two patterns $p_{ij}$ and $p_{ji}$ by comparing two sentences $s_i$ and $s_j$ depending on the order of terms in the original sentences. Also, notice that a sentence $s_i$ can generate various patterns by comparing different sentences.

We utilized a training dataset for SVM as a small corpus for pairwise pattern extraction. (The details on the dataset will be given in Sect. 3.2.1.) In the dataset we have chronological events and labels $\{(ce_1, l_1), (ce_2, l_2), \ldots, (ce_N, l_N)\}$. A CE consists of a triplet of sentence $s$, query $q$, and year $y$, i.e. $ce_i =< s_i, q_i, y_i >$.

To obtain a more generalized pattern, we normalized query $q$, year $y$, and the numbers in sentence $s$ into [QUERY], [YEAR] and [NUM] before generating a pairwise pattern. Figure 2 lists the pseudo-code for the `GetPairwisePattern` algorithm, which receives two CEs $ce_1$ and $ce_2$ as input, and returns a pairwise pattern in $s_1$ of $ce_1$. Here the `parse` function in step 1 returns all terms of sentence $s$ in a chronological event, $ce$, retaining the order of the terms in $s$, with the difference that $q$, $y$ and numbers

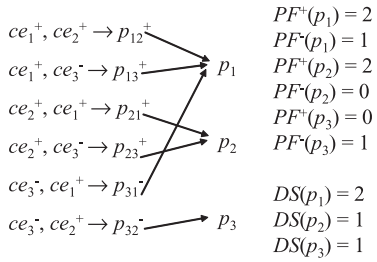


**Fig. 3** Example of pattern generation and main variables.

are normalized. Also, note that we only use a sub-sequence that contains more than $\tau$ terms to obtain meaningful pattern by filtering in step 7.

Applying the `GetPairwisePattern` algorithm to every pair of $N$ chronological events, we can obtain a pattern set, $P$, which consists of up to $N(N-1)$ patterns. However, the number of unique patterns $N_p$ should be less than $N(N-1)$ because some pairs of sentences do not contain a common sub-sequence that consists of more than $\tau$ terms, and the same pattern can be derived from different sentence pairs. The dataset also contains labels for all chronological events, and every pattern can inherit the label of its original CE. Taking into consideration the labels of patterns, we can select characteristic patterns that appear relatively high up in positive/negative sentences.

More formally, we can define positive and negative patterns as:

1. Positive Pattern $p^+$: $p \in P$ s.t. $DS(p) \geq \theta$, $PF^+(p)/\{PF^+(p) + PF^-(p)\} > N_p^+/N_p$
2. Negative Pattern $p^-$: $p \in P$ s.t. $DS(p) \geq \theta$, $PF^-(p)/\{PF^+(p) + PF^-(p)\} > N_p^-/N_p$

This is where, $PF^+(p)$ is the frequency of a positive pattern, $p$, which appears in CEs whose label $l$ is positive $(+1)$, and $PF^-(p)$ is the frequency of a negative pattern, $p$. Also, $N_p$ is the number of all unique patterns, $N_p^+$ and $N_p^-$ correspond to the number of unique patterns that have positive and negative labels, and $DS(p)$ denotes the number of unique sentences where pattern $p$ is generated. To illustrate this better, let us consider the three CEs of $ce_1^+$, $ce_2^+$ and $ce_3^-$ whose labels correspond to $+1$, $+1$ and $-1$ (Fig. 3), and whose sentences are different. The `GetPairwisePattern` algorithm can generate six patterns of $p_{12}^+$, $p_{13}^+$, $p_{21}^+$, $p_{23}^+$, $p_{31}^-$, and $p_{32}^-$ from pairs of these three CEs, and each pattern inherit the label of the CE where it was generated. Assume that the sequence of pattern $p_{12}^+$, $p_{13}^+$ and $p_{31}^-$ are the same, and can be represented as $p_{12}^+ = p_{13}^+ = p_{31}^- = p_1$ without taking into consideration the labels, and $p_{21}^+ = p_{23}^+ = p_2$, $p_{32}^- = p_3$ likewise. In this case, $PF^+(p_1)$ is 2, because $p_{12}^+$ and $p_{13}^+$ have positive labels, and $PF^-(p_1)$ is 1, because $p_{31}^-$ has a negative label. $DS(p_1)$ is 2, because $p_1$ was generated from the two sentences of $ce_1^+$ and $ce_3^-$. While $DS(p_2)$ is 1, because $p_2$ was generated from only one sentence of $ce_2^+$.

We can generate $N_p^\pm$-dimensinal binary vector $(ap_1, \ldots, ap_{N_p^\pm})$ for the feature vectors of SVM such that $ap_i = 1$ if a CE candidate contains a positive/negative pattern, $p_i$, where $N_p^\pm$ is the total number of unique positive and negative patterns. In the experiments that followed, we tried a combination of feature vectors: UG+PP, and set the parameters $\tau = 2$ and $\theta = 10$ based on a preliminary experiment.

## 3. Experiments

This section describes the details on the experiments. First, we will explain the preliminary experiment we did to determine the terms for query expansion. Second, we will discuss how we evaluated combinations of feature vectors to filter CE candidates filtering. Then, we will explain how we compared different query expansions and search strategies using the best combination of feature vectors we obtained in the previous experiment. The system was implemented in Perl on a single 1.86 GHz Intel Xeon CPU Linux server with an 8-GB memory. We employed Yahoo! API [†] as the search API. The maximum number of results per page was 50 URLs. Thus, we could obtain 100 snippets with two API calls. The experimental period was from 7 September to 19 October 2009.

### 3.1 Query Expansion Terms

Table 1 lists the query expansion terms we used in our experiment. The following sections explain the method we used for choosing the query-expansion terms in detail.

#### 3.1.1 Year Expressions

We studied the distribution of the search results for year expressions [8] to determine the year expressions for query expansion. We issued every single year expression as a phrase search query in this experiment and obtained the number of search results and 1,000 snippets per query. As a result, we observed a characteristic tendency that the number of search results of future years plunged to nearly one-thousandth from 2010 to 2025. This means that most future predictions have focused on the next one or two decades. Another interesting result for the future year is that we can see some local maximal values as sharp spikes in the distribution. Basically, these spikes appear at some round years such as 2025, 2030, 2040, 2050 and 2100.

Based on these observations, we determined the query expansion terms for future searches as all of the year expressions from 2010 to 2020 and the rounded out years of 2025, 2030, 2035, 2050 and 2100. We also determined not to use the year expressions in the Japanese style for query expansion because the number of search results was nearly one-tenth less than those for the Western style.

#### 3.1.2 Temporal Modifiers

The temporal modifiers for query expansion were determined based on the term frequency in the snippets obtained

[†]http://developer.yahoo.co.jp/

**Table 1** Query expansion terms for experiment.

| Search Strategy | Query Expansion Terms | Future | Past |
|---|---|---|---|
| Static | Year Expressions | "2010年"〜"2020年", "2025年", "2030年", "2035年", "2040年", "2050年", "2100年" | – |
| | Time Modifiers | "年までに (by the year)", "年までの (until the year)", "年で (for the year)", "年における (in the year)", "年までは (till the year)" | "年に (in the year)", "年の (of the year)", "年まで (until the year)", "年から (from the year)", "年には (by the year)" |
| | Context Terms | 予測 (prediction), 目標 (goal), 推計 (estimation), 減少 (decreasing), 増加 (increasing), 将来 (someday), 未来 (future), 上昇 (rise), 低下 (fall), ピーク (peak) | 由来 (derivation), 起源 (origin), 初 (first), 最古 (earliest), 発見 (discovery), 創設 (establish), 誕生 (birth), 発足 (launch), 創業 (found), 完成 (finish) |
| Dynamic | Year Expressions | "2021年"〜"2024年", "2026年"〜"2029年", "2031年"〜"2034年", "2036年"〜"2039年" | – |
| | Time Modifiers | – | "年は (the year is)", "年頃 (around the year)", "年までに (by the year)", "年では (in the year)", "年で (in the year)" |
| | Context Terms | 急増 (surge), 倍増 (doubling), 急減 (plunge), 半減 (halve), 予想 (forecast), 市場 (market), 倍 (times), 見込 (prospect), 達成 (achieve), 計画 (plan) | 発明 (invent), 誕生 (birth), 設立 (institution), 登場 (appear), 導入 (introduce), 最初 (first), 発売 (sale), 発表 (announce), 開催 (hold), デビュー (debut) |

by the above experiment. We counted the number of snippets containing a term sequence with a length of up to three words following year expressions (due to the structure of the Japanese language). We selected future temporal modifiers as term sequences that characteristically appear in snippets collected by future year expressions. Similarly, we selected the past temporal modifiers from snippets collected by past year expressions. Letting $SF(g)$ denote the number of snippets containing the term sequence $g$, the five steps for selecting temporal modifiers are:

1. Count $SF_{all}(g)$ within the collection of snippets for all year expressions.
2. Count $SF_{future}(g)$ within the snippets from 2010 to 2209.
3. Count $SF_{past}(g)$ within the snippets from 1800 to 2009.
4. For future temporal modifiers, select the top five term sequences based on the ratio of $SF_{future}(g)/SF_{all}(g)$.
5. For past temporal modifiers, select the top five term sequences based on the ratio of $SF_{past}(g)/SF_{all}(g)$.

Based on the above steps, we selected five future temporal modifiers, {"年までに (by the year)", "年までの (until the year)", "年で (for the year)", "年における (in the year)", and "年までは (till the year)"}. Also, we selected five past temporal modifiers, {"年に (in the year)", "年の (of the year)", "年まで (until the year)", "年から (from the year)", and "年には (by the year)"}.

### 3.1.3 Context Terms

The context terms for query expansion were determined in a manner similar to that for the temporal modifiers. Here, we selected 10 future/past context terms based on the snippet frequency of term sequences up to three words. As a result, the context terms for a future search were {"予測 (prediction)", "目標 (goal)", "推計 (estimation)", "減少 (decreasing)", "増加 (increasing)", "将来 (someday)", "未来 (future)", "ピーク (peak)", "上昇 (rise)", and "低下 (fall)"}, and the context terms for a past search were {"由来 (derivation)", "起源 (origin)", "初 (first)", "最古 (earliest)", "発見 (discovery)", "創設 (establish)", "誕生 (birth)", "発足 (launch)", "創業 (found)", and "完成 (finish)"}.

### 3.1.4 Query Expansion for Dynamic Search

We need additional query-expansion terms for dynamic search strategies because our system uses different query-expansion terms. Table 1 also lists the query-expansion terms for dynamic searches. For example, for a dynamic search with query-expansion terms for future year expressions, the system expands a query with {"2010 年"〜"2020 年", "2025 年", "2030 年", "2035 年", "2040 年", "2050 年", and "2100 年"} on the first search API call, and expands it with {"2021 年"〜"2024 年", "2026 年"〜"2029 年", "2031 年"〜"2034 年", and "2036 年"〜"2039 年"} on the second search. Here, we can expect that every single year expression from 2010 to 2040 will match the web pages containing a search query $q$.

## 3.2 Evaluation of Filtering Module

We first manually built a training dataset to evaluate the filtering module, and evaluated it with five-fold cross validation using the combination of five feature vectors explained in Sect. 2.4. As a result, the best F-measure value of 85% was achieved with a combination of all five feature vectors. We will provide more details on the training data and the method of evaluation in the following sections.

### 3.2.1 Training Dataset

We conducted future and past searches for test queries such as "robot" and "tissue engineering" to construct the training dataset for machine learning using the year expressions in Table 1 as query expansion terms. We also manually annotated a random sampling of 2,927 CE candidates $< s, q, y >$. As a result, the training dataset consisted of 1,898 positive and 1,029 negative examples.

As previously mentioned, we tried six different combinations of feature vectors: UG, UG+TM, UG+TM+CT, UG+TM+CT+SW, UG+TM+CT+SW+DY and UG+PP. We used SVM light [†] as an implementation of SVM. Precision, recall and the F-measure were estimated based on

[†]http://svmlight.joachims.org/

five-fold cross validation for the performance criteria of the filtering module. The definition of the criteria are given below.

$$Precision = \frac{PP}{PP + NP} \tag{8}$$

$$Recall = \frac{PP}{PP + PN} \tag{9}$$

$$F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

where $PP$ is the number of positive examples classified as positive, $PN$ is the number of positive examples classified as negative and $NP$ is the number of negative examples classified as positive.

### 3.2.2 Results on Filtering Efficiency

Table 2 summarizes the efficiency of filtering based on different combinations of feature vectors. The baseline in Table 2 is efficiency where the filtering module classifies all CE candidates as positive. Here, precision was 65% (=1898/2927), recall was 100% and F-measure was 79%. When we used UG as the feature vector, the F-measure increased up to 82%. Finally, the best F-measure (85%) was achieved with the vector that included all features, i.e., UG+TM+CT+SW+DY. In addition the F-measure of the text-mining approach (UG+PP) was also 85%, comparable with that of all heuristic features.

We extracted 152 positive patterns and 99 negative patterns from future CEs in the text-mining approach, and 102 positive patterns and 97 negative patterns from past CEs. There are examples of all pairwise patterns in Table 3.

There were several groups in the extracted patterns. The positive patterns from $fp_1^+$ to $fp_6^+$ for future CEs mean that both year expression $y$ and query $q$ appeared in a sentence $s$ with some particles such as "に (in)", "の (of)", "までに (until)" and a unit "年 (year)". These patterns can correspond to a combination of heuristic features TM+SW. The other positive patterns from $fp_7^+$ to $fp_{10}^+$ contain some characteristic terms for future information such as "市場 (market)", "目指す (aim)", "億 (billion)", and "万人 (thousands of people)." Therefore, these patterns contain terms of heuristic feature vector CT. The negative patterns for future CEs, from $fp_1^-$ to $fp_3^-$ contain detailed date such as "月 (month)" and "日 (day)". These patterns are not used in the heuristic approach, but they can be effective because automatically generated contents tend to be irrelevant to a query but contain detailed date expressions. The other negative patterns for future CEs, from $fp_4^-$ to $fp_{10}^-$ not only contain the target year "[YEAR] 年" but also irrelevant year "[NUM] 年". These patterns correspond to the heuristic feature-vector DY.

Likewise for past CEs, the positive patterns from $pp_1^+$ to $pp_3^+$ can correspond to a combination of heuristic-features TM+SW, containing both year expression $y$ and query $q$ with some particles such as "の (of)" and "が (is)" and a unit "年 (year)." The positive patterns from $pp_4^+$ to $pp_6^+$ can correspond to the heuristic features CT, containing characteristic terms such as "数 (number)" and "人 (people)". The remaining positive patterns from $pp_7^+$ to $pp_{10}^+$ contain the heuristic-features TM such as "に (in)" and "から (from)" and past-tense expressions such as "た (did)" or "した (did)." The negative patterns for past CEs, from $pp_1^-$ to $pp_3^-$ also contain detailed date as well as those for future CEs. The remaining negative patterns for past CEs also correspond to the heuristic feature DY containing different multiple-year expressions.

In the following experiment, we used the hyperplane derived from all of the training dataset (2,927 examples) using the features UG+TM+CT+SW+DY for the filtering module.

**Table 2** Filtering efficiency based on different features.

| | Precision | Recall | F-value |
|---|---|---|---|
| Baseline | 0.648 | 1.000 | 0.786 |
| UG | 0.798 | 0.851 | 0.824 |
| UG+TM | 0.806 | 0.856 | 0.830 |
| UG+TM+CT | 0.807 | 0.861 | 0.833 |
| UG+TM+CT+SW | 0.820 | 0.865 | 0.842 |
| UG+TM+CT+SW+DY | **0.825** | 0.873 | **0.848** |
| UG+PP | 0.822 | 0.874 | 0.847 |

## 3.3 Evaluation of the Search Module

We experimentally studied the optimal combination of query-expansion terms and search strategies based on test queries. A total of 50 test queries in Table 4 was selected

**Table 3** Example of pairwise patterns.

| ID | Positive Patterns for Future CEs | ID | Negative Patterns for Future CEs | ID | Positive Patterns for Past CEs | ID | Negative Patterns for Past CEs |
|---|---|---|---|---|---|---|---|
| $fp^+_1$ | [QUERY]*[YEAR]年に | $fp^-_1$ | [NUM]年[NUM]月*[YEAR]年 | $pp^+_1$ | [YEAR]年の[QUERY] | $pp^-_1$ | [YEAR]年*月[NUM]日 |
| $fp^+_2$ | [QUERY]の*[YEAR]年 | $fp^-_2$ | [YEAR]年[NUM]月[NUM]日 | $pp^+_2$ | [YEAR]年*が[QUERY] | $pp^-_2$ | [YEAR]年[NUM]月 |
| $fp^+_3$ | [YEAR]年*[QUERY]に | $fp^-_3$ | [NUM]月*[YEAR]年 | $pp^+_3$ | [YEAR]年*[QUERY]が | $pp^-_3$ | [YEAR]年*[NUM]日 |
| $fp^+_4$ | [YEAR]年*までに | $fp^-_4$ | [NUM]年に*[YEAR]年 | $pp^+_4$ | [YEAR]年の*は[NUM] | $pp^-_4$ | 年の*[YEAR]年には |
| $fp^+_5$ | [YEAR]年までに[QUERY] | $fp^-_5$ | [NUM]年*、[YEAR]年 | $pp^+_5$ | [YEAR]年の*数は | $pp^-_5$ | [YEAR]年から*年にかけて |
| $fp^+_6$ | [YEAR]年まで*が[QUERY] | $fp^-_6$ | [NUM]年*[YEAR]年*[NUM]年 | $pp^+_6$ | [YEAR]年*人の | $pp^-_6$ | [NUM]年*[NUM]年*[YEAR]年 |
| $fp^+_7$ | [QUERY]市場*[YEAR]年 | $fp^-_7$ | [NUM]年[NUM]*[YEAR]年 | $pp^+_7$ | [YEAR]年から | $pp^-_7$ | [NUM]年*[YEAR]年*[NUM]年 |
| $fp^+_8$ | [YEAR]年に*を目指す | $fp^-_8$ | [NUM]年から[YEAR]年 | $pp^+_8$ | [YEAR]年から*た。 | $pp^-_8$ | [YEAR]年に*年に*た。 |
| $fp^+_9$ | [YEAR]年に*[NUM]億 | $fp^-_9$ | [NUM]年～[YEAR]年 | $pp^+_9$ | [YEAR]年には*た。 | $pp^-_9$ | 年に*[YEAR]年に*た。 |
| $fp^+_{10}$ | [YEAR]年*万人 | $fp^-_{10}$ | [NUM]年*[YEAR]年まで | $pp^+_{10}$ | [YEAR]年に*ました。 | $pp^-_{10}$ | [YEAR]年に*年に*た。 |

**Table 4** Test queries for evaluating search module.

| Company | Industry | Technology |
|---|---|---|
| NEC | agriculture | robot |
| Panasonic | steel | cell phone |
| Toyota | service | internet |
| Yahoo | retailing | solar cell |
| Sonny | bank | electric car |
| Society | Economy | Education |
| immigrant | depression | elementary school |
| birthrate | inflation | high school |
| telework | poverty | university |
| pension | water resource | foreign student |
| aging society | oil price | post phD |
| Medics | Sports | Entertainment |
| tissue engineering | Olympic | movie |
| HIV | soccer | game |
| obesity | baseball | TV |
| flu | golf | electric book |
| DNA | tennis | digital contents |
| Politics | | |
| Ministory of Defence | | |
| Ministory of Health, Labour and Welfare | | |
| Ministry of Education, Culture, Sports, Science and Technology | | |
| Ministry of Economy, Trade and Industry | | |
| Cabinet Office | | |

for this experiment from ten different domains, e.g., companies, industries, technical terms, and medical terms. As a result the best efficiency for a future search was achieved by using a dynamic search with query expansion using year expression and context terms. Temporal expressions worked well for query-expansion in a past search, and we did not see any significant effect from the combination of other query-expansion terms or dynamic searches. We will report more details of the experiment in the following sections.

### 3.3.1 Query Expansion and Search Strategy

We first tried a static search for the test queries with the query-expansion terms in Table 1 to discover the basic effects of query expansion. We compared the total number of filtered CEs for 20 queries using not only a single type of query-expansion term but also combinations of them.

We next tried a dynamic search with the best combination of query-expansion terms, which could collect the most CEs in the static search. Query-expansion with the combination of year expression and context terms collected the largest number of CEs (to be explained later) for the static future search. Consequently we used this combination for the dynamic search. We compared two options for the dynamic future search: only changing year expressions in the second search and only changing context terms in the second search. However query expansion with the combination of temporal modifiers and context terms for the static past search achieved the best efficiency (to be explained later). Thus, we compared two options for the dynamic past search: only changing temporal modifiers and only changing context terms.

### 3.3.2 Result for Search Efficiency

The bar chart in Fig. 4 plots the total number of extracted and filtered CEs for 50 test queries [†]. The gray bars in the figure represent the number of CEs collected by the static search, and the black bars represent the results for the dynamic search. We used an asterisk '*' to denote a combination of different query terms. Also, the query-expansion terms in square brackets means that they were changed on the dynamic search. For example, [YE]*CT means the system used a combination of YE and CT and YE was changed dynamically at the second search API call. Note that theoretically at most 5,000 snippets were collected by 50 queries because 100 results were returned by two API calls. However the number of collected CEs could have been be more than 5,000 because multiple CEs could be extracted from a snippet. Although not all CEs retrieved in this experiment are correct, we assumed that the number of CEs can represent efficiencies under different conditions because the filtering condition was fixed in this experiment.

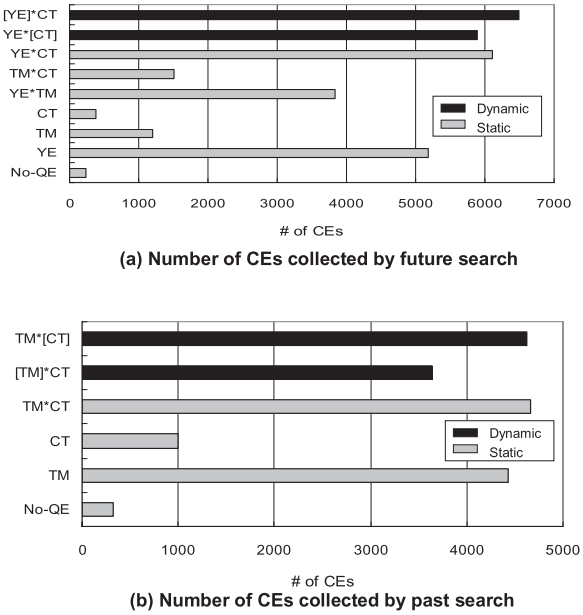


**Fig. 4** Total number of CEs collected by different query-expansion terms and search strategies.

Figure 4 (a) shows the number of CEs collected by the future search. In Fig. 4 (a), we can see that only 243 CEs for 50 queries (4.8 CEs/query) were collected without query expansion (No-QE), but 5,186 CEs (104 CEs/query) were collected with query expansion by year expressions, which is about 22 times as many CEs per query. However, query expansion by temporal expressions and context words was not as efficient in that only several hundred CEs were collected. This low efficiency indicates that even future temporal modifiers such as "年までに (by year)" or context terms such as "予測 (prediction)" could match many snippets with past years. The combination of year expressions and context terms for query expansion collected 6,168 CEs, which was the largest total in the static search.

[†]20 out of 50 queries are the same as the queries used in Sect. 3.2.1

Only changing year expressions collected 6,496 CEs for the dynamic future search, which was the largest total of all and 5.3% higher than that for the static search with the same combination of query-expansion terms. The average number of collected CEs per query between static and dynamic searches had statistically significant difference in a t-test, i.e., $p < 0.05$. This was because the changed year expressions in the dynamic search could match different CEs that were not found in the first search. However, the dynamic search where only context terms were changed was not so efficient. This is because several context terms could be included in the same snippets, and duplicate CEs could be extracted even though queries were expanded with different context terms.

The bar chart in Fig. 4 (b) has the number of CEs collected by the past search. We can see that only 319 CEs for 50 queries (6.4 CEs/query) were collected without query expansion (No-QE) in the figure, but 4,436 CEs (89 CEs/query) were collected with query expansion by temporal modifiers, which is about 14 times as many CEs per query. Compared with the future search, more CEs could be collected even without query expansion. This high efficiency of past temporal modifiers can be explained with the same reason as that for the low efficiency of future temporal modifiers.

For the dynamic past search, changing either temporal expressions or context terms could not improve the results for the static search with the same combination of query expansion terms. Actually, there were no statistically significant differences between TM, TM*CT and TM*[CT] in a t-test, i.e., $p < 0.05$. This can also be explained with the same reason that changing context terms did not work well in the dynamic future search.

### 3.3.3 Sample Results for Future/Past CEs

The bar charts in Table 5 have the examples of future CEs searched by ChronoSeeker. Note that the actual future CEs were Japanese but we translated them into English due to space limitation. Also note that there are more CEs in the near future, but we only selected the biggest cluster of CEs per year.

We found that CEs in the near future tended to be limited to countries or regions for the query "agriculture", e.g., technological issues and farming policies in different countries such as the Japanese Government's ICT strategy for agriculture and agricultural subsidies in the EU around 2013. However, in the distant far future after 2020, topics in CEs become more global and serious in association with global warming, poverty, and the shortage of water resources. By taking into account both issues in the near and distant future, we can have a long-term view of the agricultural problems.

We found for the query "robot" that South Korea is aiming to become a leading country in the robotics industry, and was actively announcing their technology roadmaps. For example, they are planning a combat robot, robot cities, and robot doctors. We could not list all their plans in Table 5, but there are other plans by South Korea regarding robot technology such as robot teachers, robot amusement parks, and robots for exploring the bottom of the ocean. In terms of combat robots, not only Korea but also the US plans to replace one third of its military forces with to robot warriors. We also found some CEs on personal robots and service robots in several marketing reports and predictions by experts. The most outstanding claim was "Humans will be

**Table 5** Example of future CEs.

| Year | Agriculture - Future |
|---|---|
| 2011 | Japanese Government will encourage ICT for medics, education and **agriculture** after **2011**. |
| 2013 | EU agrees to end **agricultural** subsidies by **2013**. |
| 2020 | By **2020**, in some countries, yields from rain-fed **agriculture** could be reduced by up to 50%. |
| 2025 | 70 % of freshwater withdrawals are for **agriculture**, by **2025** half the world's population will live in water stressed river basins. |
| 2050 | By **2050**, total average annual net investment in developing country **agriculture** would amount to USD 83 billion. |

| Year | Robot - Future |
|---|---|
| 2011 | South Korea announced a joint project worth 32.4 million dollars to develop a combat **robot** system by **2011**. |
| 2013 | South Korea to Build **Robot** Cities by **2013**. |
| 2018 | The South Korean government is hoping to release a '**robot** doctor' by **2018** which can perform surgeries. |
| 2020 | Japan aims for walking **robot** on the moon by **2020**. |
| 2050 | Humans will be marrying **robots** by the year **2050**. |

**Table 6** Example of past CEs.

| Year | Solar cell - Past |
|---|---|
| 1839 | Alexandre-Edmond Becquerel discovered the photovoltaic panel, which is the physics behind the **solar cell**, in **1839**. |
| 1941 | An American researcher, Russell Ohl, patented the first silicon-based **solar cell** in **1941**. |
| 1954 | In **1954** Bell Labs produced the first practical silicon **solar cell**. |
| 1959 | Sharp was founded in 1915, and began developing its first **solar cell** in **1959**. |
| 1975 | In **1975**, screen printing was first applied to **solar cells** for the formation of the front and rear contacts. |

| Year | DNA - Past |
|---|---|
| 1869 | Friedrich Miescher, the Swiss scientist discovered **DNA** in **1869**. |
| 1951 | Watson and Crick started researching together to decipher the structure of **DNA** in **1951** |
| 1953 | Watson and Crick discovered in **1953** that **DNA** is shaped like a double helix. |
| 1973 | In **1973**, recombinant **DNA** technology was born and the age of the "new biotechnology" came upon us. |
| 1980 | Gilbert and Sanger won the Noble prize for **DNA** sequencing in **1980**. |

marrying robots by the year 2050", which was claimed by David Levy, a British artificial intelligence researcher.

Table 6 lists the examples of past CEs. We found the discovery of photovoltaics for the query "solar cell", the first patent for a silicon-based solar cell and the main players in the early stages of the solar-cell market. We also found various topics on dye-sensitized solar cells, market size, and electromotive force in solar cells in other past CEs that are not in the Table 6. Likewise, we found some epoch-making events and technology innovations related to the query for "DNA".

## 4. Related Work

Extracting dates from text is well known as being a part of Named Entity Recognition tasks in natural language processing. Much work has recently utilized Web-search APIs to collect large corpora. Artequakt [9] can automatically collect the biographies of artists from the web. Pasca et al. [14] collected pairs of a person's name and birth year from 100 million web pages using a bootstrapping technique. Mani et al. [12] studied an algorithm that could convert relative year expressions such as "three years later" into absolute year expressions. There is also a great deal of other related work such as experience mining [5] but most of this has mainly focused on past events.

The research on prediction based on web contents has quite a long tradition. The previous attempts have mainly focused on the prediction of movements in stock prices or estimates of sales volume [3], [4], [19]. For example, Wuthrich et al. [19] proposed a method of predicting stock indices using historical news about companies and past information on stock indices as training data. Choudhury et al. [3] tried to predict changes in stock prices based on blog communication patterns. Gruhl et al. [4] demonstrated that the volume of blog postings could be used to predict spikes in actual consumer purchase decisions based on the example of books. All these work aimed at predicting the dynamics of stock prices or products' sales, while we extracted and analyzed future-related information on all sorts of events.

Some research on prediction has been done using sentiment analysis [11], [13], [15]. Mishne and Glance [13] and Liu et al. [11] applied methods of sentiment analysis to Weblog data to estimate the success of movies. Pepe and Bollen [15] investigated the public mood oncerning the future on the basis of emails submitted to futureme.org, a Web service that allows scheduled e-mails to be sent at future dates.

There has, to the best of the authors's knowledge, been relatively little work that has treated retrieval of future events. A pioneering work in future searches was done by Baeza-Yates [1]. Jatowt et al. [6] recently studied a method of extracting implicit/explicit information on the future from news archives on the Internet. Similar to this work, they also collected data by crawling through search-engine indices and analyzing collective view of future time-referenced events that were being discussed on the Web [7]. In contrast, we focused on query expansion and search strategies that could collect CEs efficiently, and we also developed a prototype system for the English language [2].

There is another research area called "prediction markets [18], which are speculative markets created for the purpose of making predictions based on the collective intelligence/wisdom of the crowd. The Iowa Electronic Market [†], and the Hollywood Stock Exchange [††]are typical examples of prediction markets. Most topics on prediction markets generally tend to be short-term forecasts ranging from several months to a year. In contrast, we focused on long-term forecasts ranging over several years.

## 5. Conclusion

This paper proposed ChronoSeeker, an on-demand search engine for future/past events that utilizes query expansion and search strategies that can collect CEs, and employs a machine-learning technique to filter out noisy CE candidates. Our experiment revealed that filtering achieved an 85% F-measure, and that query expansion could collect dozens more CEs than those without expansion.

An important contribution of this work was that we could demonstrate the feasibility of a future/past-event search engine. The system utilizes a relatively simple and computationally inexpensive way of finding CEs through search API calls and lightweight text processing, while still having a great deal of potential and usefulness. We hope that we can encourage similar kinds of studies through this work that will bring us closer to the objective of planning long-term future strategies based on future/past information on events.

In future work, we would like to treat with various time expressions, e.g. explicit and absolute expressions, detailed time expressions such as month and date. We also want to develop language independent method to see different views for the future/past events in different countries.

[†]http://www.biz.uiowa.edu/iem/index.cfm

[††]http://www.hsx.com/

### References

[1] R. Baeza-Yates, "Searching the future," ACM SIGIR Workshop MF/IR, 2005.

[2] P. Brun, H. Kawai, K. Kunieda, and K. Yamada, "ChronoSeeker: Future opinion extraction and classification," 2009 IEEE/WIC/ACM International Conference on Web Intelligence, 2009.

[3] M.D. Choudhury, H. Ssundaram, A. John, and D.D. Seligman, "Can blog communication dynamics be correlated with stock market activity?," Proc. HT2008, pp.55–60, 2008.

[4] D. Gruhl, R.V. Guha, R. Kumar, J. Novak, and A. Tomkins, "The predictive power of online chatter," Proc. SIGKDD2005, pp.78–87, 2005.

[5] K. Inui, S. Abe, H. Morita, M. Eguchi, A. Sumida, C. Sao, K. Hara, K. Murakami, and S. Matsuyoshi, "Experience mining: Building a large-scale database of personal experiences and opinions from web documents," Proc. 2008 IEEE/WIC/ACM International Conference on Web Intelligence, pp.314–321, 2008.

[6] A. Jatowt, K. Kanazawa, S. Oyama, and K. Tanaka, "Supporting analysis of future-related information in news archives and the web," 9th ACM/IEEE-CS Joint Conference on Digital Libraries, 2009.
[7] A. Jatowt, H. Kawai, K. Kanazawa, K. Tanaka, K. Kunieda, and K. Yamada, "Analyzing collective view of future, time-referenced events on the web," Proc. 19th International World Wide Web Conference (WWW 2010), pp.1123–1124, 2010.
[8] H. Kawai, A. Jatowt, K. Tanaka, K. Kunieda, and K. Yamada, "ChronoSeeker: Search engine for future and past events," Proc. 4th International Conference on Ubiquitous Information Management and Communication (ICUIMC 2010), 2010.
[9] S. Kim, H. Alani, W. Hall, P.H. Lewis, D.E. Millard, N.R. Shadbolt, and M.J. Weal, "Artequakt: Generating tailored biographies with automatically annotated fragments from the web, semantic authoring," Annotation and Knowledge Markup Workshop in the 15th European Conference on Artificial Intelligence, 2002.
[10] R. Kimura, S. Oyama, and K. Tanaka, "Automatic collection of personal histories for generating Who's Who from the web," DBSJ Letters, vol.5, no.2, 2006.
[11] Y. Liu, X. Huang, A. An, and X. Yu, "ARSA: A sentimentaware model for predicting sales performance using blogs," Proc. SIGIR 2007, pp.607–614, 2007.
[12] I. Mani, J. Pustejovsky, and B. Sundheim, "Introduction to the special issue on temporal information processing," ACM Trans. Asian Language Information Processing, vol.3, no.1, pp.1–10, 2004.
[13] G. Mishne and N. Glance, "Predicting movie sales from blogger sentiment," Proc. Spring Symposia on Computational Approaches to Analyzing Weblogs, 2006.
[14] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain, "Organizing and searching the world wide web of facts – Step one: The one-million fact extraction challenge," 21st National Conference on Artificial Intelligence, 2006.
[15] A. Pepe and J. Bollen, "Between conjecture and memento: Shaping a collective emotional perception of the future," Proc. AAAI 2008 Spring Symposium on Emotion, Personality and Social Behavior, 2008.
[16] A. Podelko, "Multiple dimensions of performance requirements," 33rd International Computer Measurement Group Conference, 2007.
[17] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995.
[18] J. Wolfers and E. Zitzewitz, "Prediction markets," J. Economic Perspectives, vol.18, no.2, pp.107–126, 2004.
[19] B. Wuthrich, D. Permunetilleke, S. Leung, V. Cho, J. Zhang, and W. Lam, "Daily prediction of major stock indices from textual WWW data," Proc. SIGKDD1998, pp.364–368, 1998.
[20] Zona Research, The Need for Speed II, Zona Market Bulletin, Issue 05, 2001.

**Hideki Kawai** received MS in Instrumentation Engineer from Keio University in 1998. In 1998, he joined Human Media research laboratories, NEC Corporation. From 2005 to 2006, he was a visiting scholar of Computer Science Department, Stanford University. Now, he is a principal researcher of C&C Innovation Research Laboratories, NEC Corporation. His research interest covers knowledge discovery and information extraction based on machine learning and text mining. He is a member of the IEEE, the ACM, the IPSJ, and the DBSJ.

**Adam Jatowt** received MS in Electronics and Telecommunications from the Technical University of Lodz, Poland in 2001. In 2005 he received PhD in Information Science and Technology from University of Tokyo, Japan. He has worked as a research fellow at the National Institute of Information and Communications Technology, Japan in 2005. From 2006 to 2009 he worked as an assistant professor and since 2010 he has been working as an associate professor at the Kyoto University. His research interests include temporal information processing, web mining, document comprehension and social bookmarking.

**Katsumi Tanaka** has been a full professor of Department of Social Informatics, Graduate School of Informatics at Kyoto University since 2001. He received BS, MS and PhD degrees in Information Science from Kyoto University, in 1974, 1976 and 1981, respectively. In 1986, he joined the Department of Instrumentation Engineering, Faulty of Engineering at Kobe University as an associate professor. In 1994, he became a full professor at the Department of Computer and Systems Engineering Department, Faculty of Engineering, Kobe University. His research interests include database theory and systems, Web information retrieval, and multimedia content retrieval among others. Prof. Tanaka is a co-author of more than 200 papers published in international conferences and journals. He is a member of the ACM, IEEE, the Database Society of Japan (DBSJ) and the Information Processing Society of Japan (IPSJ). Prof. Tanaka serves also as editorial board member of the World Wide Web Journal by Springer. He was the organizer of many international events such as DASFAA1997, FODO1998, WISE2001, APWeb2005 and ICADL2006.

**Kazuo Kunieda** is a senior manager of C&C Innovation Research Laboratories, NEC corporation. He received the B.E., M.E. and D.E. degrees in Information Science from Kyoto University, in 1987, 1989 and 1995, respectively. In 1992, he joined Kansai C&C research laboratories, NEC Corporation. His current research focus includes computational models of human behavior for improving creativity. He is a member of Information Processing Society of Japan.

**Keiji Yamada** received the B.E., the M.E., and the Dr. Eng degrees in information science from Kyoto University in 1982, 1984, and 1987. In 1987, he joined C&C information Technology research laboratories, NEC Corporation. From 1990 to 1991, he was a visiting scholar of Computer Science and Engineering Department, University of California at San Diego. Now, he is a general research manager of C&C Innovation Research Laboratories, NEC Corporation. His research interest covers knowledge processing and human communication based on pattern recognition, computer vision, artificial neural network, machine learning, human interaction, intelligent network, and socio-technology. He is a member of the IEEE and the IPSJ.