Detection of Temporal Document Features for Information Retrieval

D. M. Llidó[1], R. Berlanga[1], M. J. Aramburu[1] and A. Pons[2]

[1] Departament de Llenguatges i Sistemes Informàtics, Universitat Jaume I, E-12071 Castellón. Spain. : {dllido,berlanga,aramburu}@inf.uji.es,
[2] Departamento de Computación, Universidad de Oriente, Santiago de Cuba, Cuba aurora@app.uo.edu.cu

Abstract. In this paper we concern with the automatic generation of document temporal metadata, and how to exploit it in current Information Retrieval and Topic Detection systems. Specifically, we first propose a method to automatically detect the time references appearing in a document and to translate them into a formal time model. Then, we describe an algorithm that constructs the event-time period of a document starting from its extracted time references. Finally, we demonstrate through a clustering experiment that such generated event-time periods are as accurate as regarding all the time references of the document. This greatly simplifies the representation of the document temporality and its management within IR and databases systems.

1 Introduction

Many digital documents that currently populate the Web have a relevant temporal component. Newspaper articles, medical reports and legal texts are some examples of documents whose contents can be clearly located along time. Clearly, regarding the temporality of documents can be helpful in Information Retrieval and Knowledge Discovery tasks. In fact, the Dublin Core metadata initiative has include several elements to express several temporal aspects of documents (e.g. the creation date, the publication date, and the temporal coverage of the document). Moreover, the MUC entity task has also included the tag `TIMEX` to identify the time references within a text.

However, current Information Retrieval (IR) systems do not make use of these temporal features, mainly because they do not support a set of temporal operators [Ara01]. Additionally, these metadata need to be manually assigned, which is infeasible in applications where the flow of documents is too high. As a result, these systems only deal with the publication date, which cannot express properly the location of the document contents.

In this paper we concern with the automatic generation of document temporal metadata, and how to exploit it in current IR [Bae00] and Topic Detection systems [Pap99]. Specifically, we first propose a method to automatically detect the time references appearing in a document, and to translate them into a formal

time model. Then, we describe an algorithm that constructs the event-time period [Ara01] of a document starting from its extracted time references. Finally, we demonstrate through a clustering experiment that such generated event-time periods are as accurate as regarding all the time references of the document. This greatly simplifies the representation of the document temporality and its management within IR and databases systems.

The remainder of the paper is organized as follows. Section 2 describes the formal time model and the method for extracting time references from texts. Section 3 is dedicated to describe the different temporal features of documents. Section 4 presents two approaches to measure the similarity between documents, both taking into account the document temporal features. Finally, Section 5 is dedicated to the clustering experiment, that evaluates the retrieval effectiveness of the proposed measures.

2 Extracting Time References

The temporal sentences expressed in Natural Language usually involve the use of multiple calendar granularities. In this section we describe a time model that takes into consideration the time entities appearing in temporal linguistic sentences with supports multiple granularities. Afterwards, we describe the tool TimeExtractor, which detects all the time references from the document text, codes them according to the formal time model, and resolves the coded time references in order to obtain concrete dates or date intervals.

2.1 Time Model

Granularities. In our time model we adopt a linear time domain that is isomorphic to the set of natural numbers. Thus, each time instant is uniquely associated to a natural number, and the $\leq$ relationship represents the total order between these time instants.

Over this time line, the time model defines a set of granularities to express the different abstraction levels of the Gregorian Calendar. Specifically, we have identified the following granularities: day (i), month-day (d), week-day (x), week (w), month (m), quarter (t), 4-month (q), semester (e), year (y), decade (z), century (s) and millennium (l). The letters between parenthesis represent the code that we use to identify each granularity in the time model.

From these granularities, the Gregorian Calendar can be formally defined as follows [Bet00]:

$$\mathcal{C} = (\mathcal{G}, \mathcal{E}) \tag{1}$$

where:

- $\mathcal{G} = \{(i, \mathbb{Z}), (x, 1..7), (y, \mathbb{Z}), (z, 1..10), (s, \mathbb{Z}), (l, \mathbb{Z}), (m, 1..12), (e, 1..2), (c, 1..3), (t, 1..4), (d, 1..L_d), (w, 1..L_w)\}$, here each pair $(g, \mathcal{L}_g)$ represents a granularity g along with its domain $\mathcal{L}_g$,
- day (i) is the *terminal granularity* for the partition relationship $\overline{\wedge}$ and,

– $\mathcal{E}$ contains the rules that generate the valid values of each granularity, which are described in the following table.

Relation	Semantics
$\overline{\wedge}$	partition
$\wedge$	order
Δ	group
$\curlywedge$	finer
π	groups periodically into
$\sqsubset$	sub-granularity
$\leftrightarrow$	equivalent

Granularity Relationships

Granularity	Rule
(x,1..7)	$x \leftrightarrow i,\ \exists\ j\ \in\ \mathbb{Z},\ k = mod(j/7) + 1$
(y,$\mathbb{Z}$)	$\pi(y, i) = 400$
(z,1..10)	$\pi(\mathbb{Z}, y) = 1, 10$
(s,$\mathbb{Z}$)	$\pi(s, y) = 1, 100$
(l,$\mathbb{Z}$)	$\pi(l, y) = 1, 1000$
(m,1..12)	$\pi(m, y) = 12$
(s,1..2)	$\pi(m, s) = 6$
(c,1..3)	$\pi(m, c) = 4$
(t,1..4)	$\pi(m, t) = 3$
(d,1..L_d)	$d \leftrightarrow i,\ and\ L_d \in$ [28..31] year dependent
(w,1..L_w)	$w \leftrightarrow i,\ L_w \in$ [1..7] month dependent

Calendar generation rules

As it can be noticed, our model distinguishes between two types of granularity, namely: *relative* and *absolute* domains. Absolute domains can be always mapped into the time domain of the model, that is, each granularity domain value has associated a disjoint set of concrete time instants. Relative granularity domains need to be combined with absolute ones to define concrete time instants. The granularities *day*, *year*, *century* and *millenium* have absolute domains, and their domain is $\mathcal{L} = \mathbb{Z}$. Nevertheless, it must be pointed out that in some contexts these granularities could take relative domains (e.g. 'the second year of the century'). For this reason we introduce the equivalent granularities *day*, *month-day* and *week-day*, which help us to distinguish between the absolute domain, and the relative domains with respect to month and week, respectively.

Time Entities and Operators. We define the following time entities in our model:

– A *time point* is expressed as an alternate sequence of granularities and natural numbers :
  $T\ =\ g_1 n_1 g_2 n_2\ \ldots\ g_k n_k$
  where $g_i\ \in\ \mathcal{G},\ n_i\ \in\ \mathcal{L}_{g_i}$, and if $i < j$ then $g_j\ \overline{\wedge}\ g_i$.
  Consequently, the sequence of granularities must be ordered by the partition relationship. The order relationship $\leq$ can be defined between time points with the same granularities:
  Let $T\ =\ g_1 n_1\ \ldots\ g_k n_k$ and $T'\ =\ g_1 n'_1\ \ldots\ g_k n'_k$ be two time points, then $T\ \leq\ T'$ if $n_i \leq n'_i$ for all $1 \leq i \leq k$.
– A *time interval* is an anchored span of time that can be expressed with two time points having the same sequence of granularities:
  $I = [T1, T2]$, where $T1 \leq T2$.
– Finally, a *span of time* is defined as an unanchored directed interval of time (+ towards the future, - towards the past).
  $S = \pm ng$, where $g \in \mathcal{G}$, $n \in \mathcal{L}_g$, and $\pm$ is the direction of the span.

In order to extract and resolve time references, the systems uses the set of temporal operators showed in Table 1 [Lli01]. Following we give some examples of these operators:

```
format(y1999m1d1,ymwx)=y1999m1w1x5
format(y1999d1,ymd)=y1999m1d1
refine(y1999, m)  =[y1999m1, y1999m12]
refine(y2000m3, w) = [y2000m3w1, y2000m3w5]
refine([y2000, y2001], m) = [y2000m1,2001m12]
shift(y1999m3, +10m) = y2000m1
shift(y1998m2w2, -3w) = y1998m1w4
```

Operator	**Semantics**
$start(I)$	Starting point of I
$end(I)$	Ending point of I
$format(T, g_1 \dots g_k)$	Format T according to the specified granularity pattern
$refine(T, g)$	Refine T into a finer granularity g
$abstract(T, g)$	Abstract T up to a coarser granularity g
$shift(T, S)$	Shift T by using the span S

Table 1. Temporal Operators.

2.2 TimeExtractor

TimeExtractor is a tool that detects and analizes the temporal expressions of the document texts. This tool returns the original document with all the temporal expressions labelled with the XML tag `TIMEX` [Chi97].

In order to deal with the flexibility and fuzziness of the Natural Language, TimeExtractor defines a shallow representation language, called *CodTemp*, which includes all the elements of the formal time model as well as further symbols to capture the semantics of the linguistic expressions. For example, this language includes the code `r` to denote a time reference relative to a date, the code `R` to denote a time reference relative to an event, the code `0` to denote present time, and so on. More details of this language can be found in [Lli01].

Time Extractor is composed by two modules that work in cascade, namely: *TagTime*, which extracts the temporal expressions and codes them into the *CodTemp* language, and *ModelTimex*, which analyzes each coded expression and tries to calculate the date or date interval related to it.

The module *TagTime* makes use of a lexicon with all the grammatical elements involved in temporal expression. This lexicon is classified into three categories: *Time head nouns*, which are words associated to the calendar granularities and their domains (e.g. day, semester, June, Monday, etc.), *Quantifiers*, which are adjectives that denote some order or quantity (e.g. first, second, two, etc.), and finally *Modifiers*, which are words that denote the temporal direction and the composed time entities such as intervals and lists. All these elements have associated a semantic code from the *CodeTemp* description language. The way

TagTime recognizes the temporal expressions is similar to current Information Extraction Systems, and it is described in detail in [Lli01].

The module *ModelTimex* makes use of the relationships and operators defined in the time model in order to translate the coded time references into concrete dates. This module also uses the publication date of the document to solve the relative time expressions that refer to the reader time reference point.

The division in two modules improves the portability of this tool to other languages. Thus, to support a new language it is only necessary to rebuild the lexicon and then create the grammars that recognize the particular temporal expressions of the language. Notice that the module *ModTimex* is independent of the particular language, and it does not need to be modified.

Following we present some examples of the output of the first module:

```
<TIMEX Value='-1ny#nm#nd'> yesterday </TIMEX>
<TIMEX Value='IO-wp'> from two weeks ago </TIMEX>
<TIMEX Value='#Oy#m6#d1'> first of June </TIMEX>
<TIMEX Value='34yp'> 34 years </TIMEX>
<TIMEX Value='I#y1992'> from 1992 </TIMEX>
```

These time expressions are processed by the second module, which produces the final output (here the publication date is 01/06/1999):

```
<TIMEX type='DATE' Value='y1999m05d31'> yesterday </TIMEX>
<TIMEX type='DATE' Value='[y1999m5w3,19990531]'> from two weeks ago </TIMEX>
<TIMEX type='DATE' Value='y1999m6d1'> first of June </TIMEX >
<TIMEX Value='34yp'> 34 years </TIMEX>
<TIMEX type='DATE' Value='[y1992,y1999]'> from 1992 </TIMEX>
```

3 Document Temporal Features

In order to index the input documents in an Information Retrieval System (IRS) [Bae00], we need to extract all the features that will be used in the retrieval engine. Traditionally, an IRS represents each document with the list of terms weighted by their frequency of appearance in the document text (TF vector). This weight is usually modified to take into consideration the overall frequency of each term in the collection (IDF).

In our approach, we also need to represent the temporal features extracted from the texts. Similarly to the term vector, we can represent them as a list of time entities weighted by their frequency in the document text, which is called *Vector of Weighted Time Entities.*

Since many IRS and database models do not support the representation of multiple dates, we have also introduced in the document representation the *event-time* period [Ara01]. How can we obtain it from the set of extracted dates?

To determine the event-time period of a document we can start from the following hypothesis:

– Events are published few days after they happened. So, time references far away from the publication date are references to other related events.

– Interval references are less important as greater the duration is.
– Long duration events (e.g. 'a war', 'a trial') are usually referenced by some relevant days that are not contiguous. In this way, to determine their event-time periods we must allow some gaps between the relevant dates appearing in the article.

Algorithm 1 Event Time Algorithm

Require: pd^i, D^i, I^i, $threshold$, gap, $maxdist$
{pd^i: publication date ; D^i: list of extracted points entities ; I^i: list of extracted intervals entities ; $threshold$: lowest frequency for a relevant time entity ; gap: allowed days-distance between relevant time-entities ; $maxdist$: maximal days-distance between two proximal dates}
Ensure: $Event\_Time$
for all $[f_1, f_2] \in I^i$ **do**
  if $dayDistance(f_1, f_2) \leq maxdist$ **then**
    add to D^i all dates between f_1 and f_2
    delete $[f_1, f_2]$ from I^i
  else
    add to D^i the dates f_1 and f_2
  end if
end for
$F0 = '\ '$; $cont = 0$; $F1 = ''$; $FI = ''$; $FF = ''$
for $F1 \in D^i$ **do**
  if $F0 == '\ '$ **then**
    continue
  else
    if $dayDistance(F1, F0) < gap$ **then**
      $cont+ = D^i[F1]$; $FF = F1$
    else
      $I^i[FI, FF] = cont$
      $F0 = F1; FI = F1; FF = F1; cont = 0$
    end if
  end if
end for
$Event\_Time = ExtractMostRelevantInterval(I^i, maxdist, pd^i, threshold)$
{Obtain most relevant interval near to the pd^i}
if not $Event\_Time$ **then**
  $Event\_Time = [pd^i - 1, pd^i]$
end if

The algorithm we propose to generate document event-time periods (see Algorithm 1) takes into consideration these hypothesis. Basically, the algorithm determines the most relevant interval near the publication date. For this purpose, the algorithm first builds a list of intervals with the contiguous time entities of the document, allowing some gaps between them. To determine the relevance of each interval, the algorithm uses the frequency of the time entities that overlap

with each interval. Only the intervals with a relevance greater than a predefined threshold will be considered by the algorithm.
Summarizing, in our approach each document d^i is represented with the following features:

- the publication date: pd^i,
- the vector of weighted terms: $T^i = (TF_1^i, \ldots TF_n^i)$, where TF_k^i is the frequency term t_k in the document d^i,
- the vector of weighted time entities: $F^i = (f_1^i : TF_{f_1^i}, \ldots f_m^i : TF_{f_m^i})$, where $TF_{f_j^i}$ is the appearance frequency of the time entity f_j^i in the document d^i,
- and the *event-time period:* et^i.

4 Time Similarity Measures

Information Retrieval Systems as well as Topic Detection ones require the definition of a document similarity measures. The IR systems apply it, to rank query results according to the similarity value between queries and documents. The Topic Detection Systems applies it to cluster together those documents that report about the same events.

Traditionally, IRS and TDT systems uses the cosine function over the document term vectors as the similarity measure, which is as follows:

$$S_t(d^i, d^j) = \frac{\sum_{k=1}^{n} TF_k^i \cdot TF_k^j}{\sqrt{\sum_{k=1}^{n} TF_k^{i^2}} \cdot \sqrt{\sum_{k=1}^{n} TF_k^{j^2}}} \tag{2}$$

In our approach, we must also regard the similarity between the other document features. Specifically, we consider that two documents that report about a same event must be similarity in both, their terms and time features. For this purpose, we propose two similarity measures that mainly differs from the selected temporal features:

- **Time-Cosine Similarity** [Pon02]: which uses all the time entities that appears on the text. To define a global similarity measure between two documents d^i and d^j applying a temporal similarity S_f measure, we propose the following weighted measure:

$$S(d^i, d^j) = W_t S_t(d^i, d^j) + W_f S_f(d^i, d^j) \tag{3}$$

  where W_t and $W_f \in [0, 1]$ represent the weight of the term and dates vector respectively.

  The temporal similarity S_f is defined as follows:

$$S_f(d^i, d^j) = \frac{\Sigma_{k=1}^{m_i} TF_{f_k^i} \cdot TF_{s(f_k^i, d^j)} \cdot g(F_k^i, s(f_k^i, d^j)) + \Sigma_{k=1}^{m_j} TF_{f_k^i} \cdot TF_{s(f_k^j, d^i)} \cdot g(F_k^j, s(f_k^j, d^i))}{(2 + \mid m_i - m_j \mid) \cdot \sqrt{\sum_{k=1}^{m_i} TF_{f_k^i}^2} \cdot \sqrt{\sum_{k=1}^{m_j} TF_{f_k^j}^2}}. \tag{4}$$

  where:

- m_i is the amount of time entities appearing in the document d^i ,
- $s(f_k^i, d^j)$ returns the closest date in d^j to f_k^i according to the Minkowsky distance,
- and the function g is defined as follows:

$$g(f_1, f_2) = \begin{cases} 1 : if\ f_1 = f_2, \\ 0.8 : if\ d(f_1, f_2) = 1, \\ \frac{1}{\sqrt{d(f_1, f_2)}} : otherwise. \end{cases} \quad (5)$$

– **Event-Time Similarity**: which uses the event-time period associated to each document. To define a global similarity measure between two documents d^i and d^j, we propose the following measure:

$$S(d^i, d^j) = \begin{cases} S_t(d^i, d^j) :\ If\ d(d^i, d^j) > \beta_{et} \\ 0 : otherwise. \end{cases} \quad (6)$$

where β_{et} is a temporal threshold over the event-time period.

Both measures uses the Minkowski distance [Ich94] to compare date intervals. It is defined as follows:
Let f_1 and f_2 be two intervals, then

$$d(f_1, f_2) = |f_1 \oplus f_2| - |f_1 \otimes f_2| + \rho \cdot (2 \cdot |f_1 \otimes f_2| - |f_2| - |f_1|) \quad (7)$$

where :

– $f_1 \oplus f_2$ is the union interval,
– $f_1 \otimes f_2$ is the intersection interval,
– and $|f|$ is the number of days between the extremes of the interval.

To evaluate the Minkowsky distance over a date, we translate the date f_i into the interval $[f_i, f_i]$. It can be noticed that if f_1 y f_2 are dates, then Minkowsky distance return the days between f_2 y f_1. We take the value 0.2 for ρ [Pon02].

5 Evaluation

In this section we present the experimental results for our document representation model. For this evaluation, we have implemented the Single-Pass clustering algorithm, which has been widely adopted in on-line Topic Detection Systems because of its simplicity and efficiency [Yan98b].

Briefly, this algorithm puts each incoming document into the most similar already constructed cluster such that its similarity is greater than a given threshold. In this case, the cluster is updated according to the new document. If there not exists such a cluster the document becomes a new cluster.

The evaluation corpus comprises a set of news articles of June 1999 published on 'El País Digital'. From this collection of 553 articles, we have manually identified 246 topics, from which 80 are non-unitary.

The effectiveness of the cluster algorithm is measured using the F1-measure between each manually labelled topic i and each system-generated cluster j. In order to measure the global performance, we firstly associate to each topic the most similar system-generated:

$$\sigma(i) = argmax_j\{F1(i,j)\} \tag{8}$$

Then, we define an overall F1-measure weighted with the size of the groups [Pon02]:

$$F1 = \frac{1}{N_{docs}} \sum_{i=1}^{N_{topics}} n_i \; F1(i, \sigma(i)) \tag{9}$$

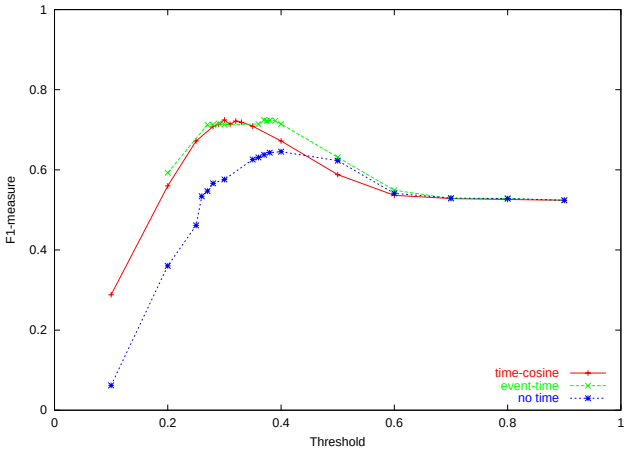

Fig. 1. Overall F1 for all topics

The optimal temporal threshold for the Time-Cosine measure is $\beta_f = 0.14$, whereas for the Event-Time one is $\beta_{et} = 0.36$. Figure 1 presents the evolution of the global F1-measure with respect to the term threshold β_t, keeping the temporal threshold constant with its optimal value. In the graphic we can notice that the curves of the Event-Time and Time-Cosine similarities are very similar. In both cases we obtain better results than using only the term similarity ('no time' curve). This latter case can be compared with the results obtained in [Yan98b], since all the articles of the collection are published in a time window of one month. As a result, we think that the use of the proposed temporal features could improve notably the results of current TDT systems.

Finally, these results also indicate that the use of the event-time period can be as effective as using all the time entities, but with the advantage that it simplifies the representation of the documents, and it reduces considerably the cost of the clustering algorithm.

6 Conclusions

In this paper we have analyzed different ways to express the temporality of documents. Firstly, we have proposed a method to extract time entities from document texts. Then, we have presented an algorithm to automatically obtain the event-time period of a document taking into account the extracted time entities.

We have shown how these temporal features can be used in Information Retrieval and Topic Detection systems, by introducing two new similarity measures between documents: one for the list of extracted time entities (Time-Cosine measure), and another for the event-time periods (Event-Time measure).

The obtained results for a clustering experiment demonstrate that both measures improve the retrieval effectiveness. Moreover, the Event-Time measure has a similar effectiveness to the Time-Cosine one. This indicates that the generated document event-time periods are good representations of the temporal coverage of the document, and they can be efficiently used in Information Retrieval and database systems.

References

[Ara01] M. J. Aramburu and R. Berlanga. "A Temporal Object-Oriented Model for Digital Librares of Documents". Concurrency: Practice and Experience, Vol. 13 (11), 2001.

[Bae00] R. Baeza-Yates and B. Ribeiro-Neto. "Modern Information Retrieval". Ed. ACM-Press, 2000.

[Bet00] C. Bettini, S. Jajodia and X. S. Wang. "Time Granularities in Databases, Data Mining, and Temporal Reasoning", Springer-Verlag, 2000.

[Chi97] N. Chinchor. "MUC-7 Named Entity Task Definition". September 1997. `http://www.muc.saic.com/proceedings/ne_task.html`

[Lli01] D. M. Llidó, R. Berlanga and M. J. Aramburu. "Extracting temporal references to automatically assing document event-time periods". In Proc. of DEXA 2001, Springer-Verlag, Munich, 2001.

[Pap99] R. Papka. "On-line new event detection, clustering and tracking". PhD Disertation, Departament of Computer Science. University of Massachusetts, 1999.

[Pon02] A. Pons, R. Berlanga and J. Ruiz-Shculcloper. "Temporal-Semantic clustering of news for Event Detection and Tracking". 2nd Workshop on Pattern Recognition in Information Systems, Alicante, 2002.

[Ich94] M. Ichino and H. Yagushi. "Generalized Minkowski Metrics for Mixed Feature-Type Data Analysis". IEEE Transactions on Systems, Man and Cybernetics, Vol. 24 (4), 1994.

[Yan98b] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald and X. Liu. "Learning approaches for Detecting and Tracking News Events". In Proc. of ACM/SIGIR, 1998.