# FORMAL GRAMMAR FOR THE NORMALIZATION OF RELATIVE TEMPORAL EXPRESSIONS

Leandro Gallina[1], Renata Galante[1] and Carina F. Dorneles[2]
[1]*Universidade Federal do Rio Grande do Sul, UFRGS*
[2]*Universidade Federal de Santa Catarina, UFSC*

## ABSTRACT

Traditional search engines use only a small part of the Web's temporal information, and remain focused mostly on the page's publication date. In this scenario, an absolute temporal expression used as keyword query does not retrieve documents containing relative temporal expressions. In this paper, we propose an approach for normalizing relative temporal expressions in natural-language text. Our approach is based on a formal grammar to identify the temporal expressions and convert them to absolute dates. In order to demonstrate the effectiveness of our approach, we have executed a set of experiments over corpora of documents with relative temporal expressions. The evaluation shows successful results of precision (82.3%) and recall (80.2%) for our approach.

## KEYWORDS

Information retrieval, text analysis, formal grammars, and temporal expressions.

## 1. INTRODUCTION

Search engines have advanced to allow users to get increasingly better results for their queries. Most commercially available search engines such as Google and Yahoo! are specially focused on the syntactic information available on Web pages. Usually, they use only a small part of the Web's temporal information, and remain focused mostly on the page's publication date. However, there is a great deal of temporal information available on the Web that could be inferred from temporal expressions.

Web pages often contain relative temporal expressions, such as `“today”`, `“three years ago”`, `“next year”`, and others. Search engines have just supported the web page's crawling date, discarding temporal expressions that appear as content in the documents (JIN et al., 2008). It may be useful sometimes: for instance, pages with the `“I Was Born About Ten Thousand Years Ago”` expression are indeed relevant if the user is searching for the music with name by `Elvis Presley`. However, the expressions contained in web pages can refer to events that happened in the past or will happen in the future. It means that to answer the user query, it is necessary to compute value dates in order to get the correct information. In this case, we need to get the relevant information for the user. We have named these expressions as **relative temporal expression**.

For instance, consider a web page extracted from the Internet[1] and dated of `February 2nd, 2005` that contains the following text: `“…in the last year alone, the United States has added 2.3 million new jobs”.` From the web page's date and the relative temporal expression `“last year”`, we can infer that the United States added 2.3 million new jobs in 2004. However, if a user submits a query like `“United States new jobs 2004”` to a conventional search engine, it may not bring the desired information; instead, it will insist on search for web pages that were originally written with the token `”2004”`. A news page might contain the desired data, but it does not contain the token `“2004”`. In this case, the page will not be found. This is often the case because news pages may use relative temporal expressions rather than the absolute date itself.

---

[1] http://www.americanrhetoric.com/speeches/stateoftheunion2005.htm

In this paper, we propose an approach for normalizing relative temporal expressions in natural-language text. Our approach is based on a novel formal grammar specially designed for the English language. The grammar aims to identify the temporal expressions and convert them to absolute dates. To the best of our knowledge, this is the first time that such grammar has been proposed for the resolution of temporal expressions. In order to demonstrate the effectiveness of our approach, we implement a parser to perform the grammar, created through the usual parsing tools such as Flex/Lex and Yacc/Bison (LEVINE et al., 1992). We have also executed a set of experiments over corpora of documents with relative temporal expressions. The evaluation shows successful results of the precision (82.3%) and recall (80.2%) for our approach.

The rest of this paper is structured as follows. Section 2 discusses related work. Section 3 presents our approach for supporting relative temporal search while Section 4 specifies in details the proposed grammar. The experiments that demonstrate the effectiveness of our approach are described in Section 5. The main ideas of this paper, and future work, are summarized in Section 6.

# 2. RELATED WORK

In order to appropriately solve relative temporal queries, we have proposed a formal grammar used to identify relative temporal expressions and then convert them to absolute dates. For clarifying our explanation, we have separated the related work in two parts. First, other approaches for relative temporal expression identification are addressed. Second, we present tools that perform date standardization.

## 2.1 Relative Temporal Expressions Identification

Several approaches have been proposed for answering queries that contain relative temporal expressions in both query and text results (ARIKAN et al., 2009; MANI; WILSON, 2000; SAQUETE; MARTÍNEZ-BARCO, 2000; ZHANG et al., 2008). Mani and Wilson (MANI; WILSON, 2000) presents a temporal annotation scheme for representing dates and times in temporal expressions, for the resolution of time expressions. Unlike our work, their system consists of a tagger that uses a variety of hand-crafted and machine-discovered rules. TOB (*Timely Ontologies for Business Relations*) (ZHANG et al., 2008) is a system for temporal expressions recognition and their reference resolution, based on a proposed temporal model. TOB has two different units: *(i)* page-level inference of absolute dates from relative temporal expressions, and (*ii*) an information extraction unit, which extracts and stores temporal facts in ontologies. In summary, TOB is a suite of methods for extracting temporal relations from text and storing them on ontologies.

Another information extraction method is presented in (ARIKAN et al., 2009), which uses an approach based on language models to allow the user to search for data in absolute temporal expressions. For instance, if a document contains the expression "*between 1992 and 2000*", the system will be able to search for information regarding the year of 1997. The work of (SAQUETE; MARTÍNEZ-BARCO, 2000) presents a formal grammar for the recognition of temporal expressions in the Spanish language and the resolution of its temporal reference. This work also solves linguistic coreferences such as *"dos días antes"* (two days before), *"la semana anterior"* (the previous week), and others, all in the Spanish language.

## 2.2 Date Standardization Tools

Dates in a document can be represented in a variety of ways. For example, the same date can be represented as "February 1st, 2011", or "02/01/2011", or other ways (XXXXX). Several approaches have been proposed to normalize dates that appear in a document (MAYNARD; CUNNINGHAM, 2003; SUCHANEK et al., 2006; VERHAGEN; PUSTEJOVSKY, 2008). Annie (MAYNARD; CUNNINGHAM, 2003) is an information extraction tool that uses named entities to extract temporal information. Annie generates an XML file with annotated entities in a format specific to the Annie project. GUTime is a temporal annotation tool from the TARSQI package (VERHAGEN; PUSTEJOVSKY, 2008). GUTime generates an XML document with annotated temporal expressions according to the TimeML format. These annotations can be manipulated using XPath, XQuery, or any other XML processing language. GUTime annotates and normalizes relative temporal expressions with regard to the document publication date. Leila (SUCHANEK et al., 2006) is an information extraction tool based on a link grammar (SLEATOR; TEMPERLEY, 1991) (sometimes called dependency grammar) to extract facts from natural-language text. Leila extends the functionality of a Link Grammar Parser to generate a graph-based representation between the words of a given sentence. The fact extraction uses statistical learning algorithms such as SVM (CHERKASSKY; YUNQIAN, 2004). We have adopted Leila in order to normalize temporal expression, since it represents all dates in the document in a standard format. Thus we are free from resolving date ambiguities, and our system does not have to implement parsing for all date formats (such as "January, 26th" or "01/26"). For this reason, we explain Leila in more details.

The date standardization step of Leila is twofold. First, Leila normalizes all dates to the standard format of `@YYYY_MM_DD@`. Second, Leila removes HTML tagging, since the user is only interested in the content of the document (where the relative temporal expressions are found). Leila standardizes dates as follows:

- All dates whose year, month and day are known is represented in the `@YYYY_MM_DD@` format. For instance, "March 31st, 2011" is normalized as `@2011_03_31@.`
- When only the year and the month are known (i.e. the day is not known), the date is normalized to `@YYYY_MM_#@` format. For instance, "December, 2011" is normalized as `@2011_12_#@`.
- Year posed alone is normalized as `@YYYY_#_#@`. For instance, "2011" is normalized as `@2011_#_#@`.
- Decades are represented in the same way of year `@YYY#_#_#@`. For instance, the "1980s" are normalized as `@198#_#_#@`.
- When only the day and the month are known (i.e. the year is not known), the date is represented as `@#_MM_DD@`. For instance, the expression "February 11th" is normalized as `@#_2_11@.`
- When only the month is known, it is replaced with `"MONTH01"` for January, `"MONTH02"` for February, and so forth.
- A complete date such as "Friday December 17th, 1999" is normalized as `"Friday @1999_12_17@"`. This normalization is important to avoid the computation of the token `"Friday"` as a separate date.

From the above output generated by the pre-processing step, we propose a grammar that builds upon the constructs already provided by Leila. For instance, the `MONTH` constructs provided by Leila are replaced with the correct year and month calculated after normalization. However, our formal grammar is specified in order to parse tokens in the way they are generated by the Leila pre-processing step.

## 3. SUPPORTING RELATIVE TEMPORAL SEARCH OVERVIEW

In this section, we present our approach for supporting relative temporal search. We propose an approach for normalizing relative temporal expressions in natural-language text. Our approach is based on a formal grammar for English language to identify the temporal expressions and convert them to absolute dates. More specifically, we convert relative temporal expressions to absolute dates by inferring from the publication date.

The main feature of our approach is a novel grammar for the normalization of relative temporal expressions that occurs in English language. To the best of our knowledge, there have been no previous attempts to use formal grammars for the resolution of temporal expressions in English. The proposed grammar (Section 4) is used to parse text in natural language form, which can be any kind of document or news item. In each document, we infer the publication date; then, based on this publication date, each relative temporal expression is converted to its equivalent absolute date. The formal grammar is used to parse all relative temporal expressions found in each document.

Our approach is separated into three main parts:

1. **Date Standardization**. The HTML tags are removed and the dates are standardized.
2. **Publication Date Identification**. The page is analyzed in search of the web page's date.
3. **Grammar Parse.** The grammar finds the relative temporal expressions, such as "today", "tomorrow", "two years ago", and others. Then, each temporal expression is normalized, based on the publication date, to an absolute date.

The first module, **1 – Data Standardization**, is responsible to convert all dates from a document to a standard format. It makes it easier to identify the publication date and pose queries later. As previously pointed in related work, we have chosen Leila (SUCHANEK et al., 2006) in order to extract relative temporal expressions.

The second module, **2 – Publication Date Identification**, is used as the reference point for the normalization of relative temporal expressions. Correctly identifying the publication date is very important for the final result: if the publication date assumed for the document is wrong, then all else fails, since the normalized dates will likely be wrong as well.

The first source for the publication date is the content of the web page. The publication date may often be found near keywords as `“published”`, `“posted”`, or `“publication date”`. In the case of a web page that allows user comments, it is important to avoid confusing the page's date with the comments' date. If the publication date is not found in the text of the page, we analyze the page's metadata. If there is no metadata with the page's date we resort to the URL.

In the following section, we specify in detail the grammar module (**3 – Grammar**) once that represents the main contribution of this paper.

## 4. FORMAL GRAMMAR FOR RELATIVE TEMPORAL EXPRESSIONS

In this section we present a novel grammar for the resolution of relative temporal expressions. First, we show a briefly example for the use of our grammar. Then, we introduce a general view of the grammar. After, we describe in details the main components of the grammar. Finally, we point key decisions and limitations.

Table 1 exemplifies expressions that may occur in a given document and the correspondent replace that we are proposing. In this table, we assume that the publication date is April 1st , 2011.

Table 1. Relative temporal expression examples.

| Relative temporal expressions | Normalized relative temporal expressions |
|---|---|
| tomorrow | April 2nd, 2011 |
| *last Tuesday* | March 29th, 2011 |
| next month | May 2011 |
| Three years ago | 2008 |

The main rules of our grammar generate the well-formed temporal expressions that are normalized in the sequence. Figure 1 specifies the main components of the grammar. Due to space limitations, we explain in the following the most important components the grammar.

```
temporal_expression → today
        | tomorrow
        | yesterday
        | month_ago
        | years_ago
        | last_weekday
        | next_weekday
        | weekday
        | last_year
        | next_year
        | day_month
        | word_month
        | date_without_year
```

Figure 1. Relative temporal expression grammar.

The resolution of a *one-word expression* is very simple. For instance, consider the `“yesterday”` expression. This expression must be replaced by the resulting of adding `-1` (negative one) day from document publication date. The resolution of `“today”` and `“tomorrow”` is analogous (by adding zero and one day, respectively). Observe the rule in the following.

```
yesterday → ”yesterday”
      addDays(documentDate, -1)
```

An expression such as `“next Thursday”` is resolved by the `next_weekday` rule. To get the next occurrence of a given weekday, we define the `getNextWeekday()` function. Observe both the rule and the function in the following. The `“last Friday”` expression is analogously resolved.

```
weekday → ”Sunday” | “Monday” | “Tuesday”
         | “Wednesday” | “Thursday” | “Friday” | “Saturday”
token_next → “next”
next_weekday → token_next  weekday
    getNextWeekday(documentDate, $2);
```

Expressions such as `“five months ago”` are resolved by the `months_ago` rule. We define the `addMonths()` function to add (and subtract) months from the current date. Observe both the rule and the function in the following. The `“three years ago”` expression is analogously resolved by the `years_ago` rule. We use `Lex` to attribute to the token `number` the value of a number in the text, including numbers written from `“one”` to `“ten”.`

```
token_month → “month” | “months”
token_ago → “ago”
months_ago → number token_month token_ago
    addMonths(documentDate, -$1);
```

When both *month and day are known*, but the year is unknown, we decided to use a regular expression to translate `@#_MM_DD@` to `date_without_year` token. Then, the last occurrence of such date (relative to the document date) is calculated using the `getDateWithoutYear()` function. This is resolved by the `date_without_year` rule. Observe both the rule and the function in the following.

```
date_without_year → [@][#][_][0-9]1,2[_][0-9]1,2[@]
    getDateWithoutYear(documentDate, $2);
```

When a token, such as `month (MONTH01, MONTH02, …, MONTH12)`, appears in the text preceded by a *number*, this indicated that the text has a date in the `“15 December”`. This means that we must get the last occurrence of the date, in the same way of the previous rule. Observe both the rule and the function in the following.

```
day_month → number  month
    getDateWithoutYear(documentDate, $1, $2);
```

If the `month` token is preceded by a *word*, the system prints the word and calculates the previous occurrence of the month using the `getDateByMonth()` function. Observe both the rule and the function in the following. For instance, this rule is applied in a sentence such as `“the vaccine was discovered` **`on December`**`”.`

```
word_month → word month
    printf($1); getDateByMonth(documentDate, $2);
```

## 4.1 Running Example

Here we present a complete example in order to illustrate the application of our approach. We extracted a web page from the Guardian[2], which is a news item with relative temporal expressions. We will exemplify the usage of our approach with the web page title. The title is:

`“UK arms companies visited Tripoli three months ago”`

The web page publication date is inferred as `February 27th, 2011`. The grammar parser pursues into the text provided. All parts `”UK arms companies visited Tripoli”` are identified by the grammar as **word**. It means that all words are transcribed as is to the output. Then, the grammar parser

---

[2] http://www.guardian.co.uk/world/2011/feb/27/libyan-arms-fair-attended-by-uk-firms

identifies the `“three months ago”` expression, which is parsed as a **number** followed by the tokens `months` and `ago`. In this moment, the system parses this expression, by identifying it can be generated by the `months_ago rule`. The **number** is converted to its integer representation (i.e 3 number). Then, we use the `addMonths()` function to subtract three months from the publication date from `February 27th, 2011`. The result is `November 2010`. Notice that the expression `“three months ago”` does not allow one to know exactly to which day of `November 2010` the web page refers. The exact day is declared as unknown using the standard notation `@YYYY_MM_DD@`. The result is `@2010_11_#@`. This date is produced as output, replacing the expression `“three months ago”` in the source document.

Notice that our choice to normalized date output in the notation of `@YYYY_MM_DD@` is somewhat arbitrary. It would possible to perform a post processing step and convert all dates back to an English readable form. However, since our purpose is to normalize relative temporal expressions so that they can be found through a search engine, it is up to the search engine to look for dates in the produced format.

## 4.2 Key Decisions and Limitations

In this section we point the key decisions that we have adopted in order to implement our grammar parse. The result is a grammar parser in C, which takes a text file as input and replaces well-formed relative temporal expressions with their equivalent absolute dates. It is important to notice that the grammar parse has been implemented using Lex and Yacc (LEVINE et al., 1992). The version of Lex used is 2.5.35. The version of Yacc used is GNU bison 2.4.1.

The main decision taken into account during the grammar definition phase was to choose if a relative temporal expression is related to past or future occurrence. For instance, consider `“Friday”` expression. Does the expression refer to the `”next Friday”` or to the `“previous Friday”`? The same ambiguity arises when there is a date such as `“January 13`$^{th}$`”`. We have decided that these relative expressions should be replaced by the previous occurrence of the given date. The documents used on our experiments include news items and reports, which mostly refer to events in the past (KOEN; BENDER, 2000). In the experiments section, we will also discuss the performance related to this decision.

The adopted granularity is day. It means that expressions that involves hour, minutes and seconds are note treated. This decision does not affect our grammar. Experiments section (Section 5) demonstrates that the most documents contains data with maximum granularity as a day. Otherwise, our grammar is easily extensible for adding new granularities.

It is important to emphasize that the scope of our grammar is English documents. We believe that English is more frequent and multilingual approaches could be easily adopted in order to apply our approach in another languages.

We highlight that our approach currently treats relative temporal expressions considering the publication date of documents. Coreferences (relative expressions to dates in the middle of the text) are not normalized. For instance, consider the following sentence: `“On Dec. 2nd, 2008, the President (...) Then, the next day, (...)”`. Our approach considers the publication date instead of referring to the day after December 2nd, 2008. We intend to solve this gap in a future work.

# 5. EXPERIMENTAL EVALUATION

In this section, we describe the experiments we have performed in order to evaluate the effectiveness of our approach. The focus of the experiments is relative temporal expressions identification. We manually check all relative temporal expressions in the documents. In each document, we manually identify all the relative temporal expressions. Then, we run the implementation of our approach and then we check if the relative temporal expressions were correctly found and processed. We collected 195 news pages from the 1994 LA Times archive. In order to analyze the results of experiments, we have evaluated them under the following aspects: *recall* and *precision*. Recall and precision are well known quality measure metrics in IR community (BAEZA-YATES; RIBEIRO-NETO, 1999).

## 5.1 Results and Discussion

Table 2 shows the main results. We identified 777 relative temporal expressions; 623 expressions were correctly normalized; 134 expressions were incorrectly normalized; and 20 expressions were not treated. The precision result was 82.3% whereas the recall result was 80.2%.

Table 2. General results.

| Total | Found | Normalized Correctly | Normalized Incorrectly | Untreated | Precision | Recall |
|---|---|---|---|---|---|---|
| 777 | 757 | 623 | 134 | 20 | 82.3% | 80.2% |

Overall, we obtained good results in terms of precision and recall in our experiments. Considering the achieved results, here we analyze the choices made during the grammar design phase that can have affected both precision and recall values. First, some temporal expressions were not being followed by any word that could indicate whether it refers to the past or to the future. We decided to use these dates in the past. In most cases, these expressions indeed refer to the past, improving the precision of our approach. However, in a few cases, they indicate dates in the future, decreasing the precision. A future solution could be to analyze the tense verb to decide whether the date refers to past or future.

Second, all occurrences of dates in the text were treated as the last occurrence of such date. For instance, in the following sentence: ″Dubov had reported Bissett missing Dec. 20″. Our approach identified “Dec. 20” data as a reference to the last occurrence of this date. In this context, it was manually identified as correct. It is important to notice that this case is more frequent. However, in a few cases, dates can be used as nouns along the text. For instance, in the following sentence: “The victims of September 11, whom we will be remembering in a few days' time”, the ″September 11″ expression refers to the events that happened specifically on September 11, 2001. It could be argued that it is a noun, rather than a relative temporal expression. Our approach processed such as data, resulting in errors. One possible solution for this problem would be to create a dictionary of expressions that are not to be processed as temporal expressions.

Some errors are due to Leila's pre-processing of the text. For instance, in Europarl (http://www.statmt.org/europarl/) we found the following sentence: “In the last week of February, 39 instances of illegal degassing were identified”. Leila’s pre-processing step resulted in: “In the last week of @#_2_39@ instances of illegal degassing were identified”. Since Leila interpreted that “February, 39” was a date without year, our approach attempted to find day 39th of February of that, which resulted in a wrong date. Another similar error occurred when regular words were confused with temporal expressions. For instance, in the sentence: “May I inform you, my honourable friends, that...”. In this case the “May” word is not a name of the month.

As for the untreated expressions, most of them arise from temporal coreferences, where a temporal expression depends on another date that appears on the text. Examples of coreference are expressions like “the previous month”, “the following day”, and others. These expressions have not been not treated. However, we notice that these expressions have appeared very scarcely in the experiments. Also untreated were expressions that denote holidays, such as “Christmas Day” or “Thanksgiving”. These cases result in the number of untreated expressions, that reduce slightly the recall of our approach.

## 6. CONCLUSION

This paper, we presented a new approach for normalizing relative temporal expressions in natural-language text. We have specified a novel formal grammar to identify the temporal expressions and convert them to absolute dates. In order to demonstrate the effectiveness of our approach, we have executed a set of experiments over corpora of documents with relative temporal expressions. The evaluation shows successful results of the precision (82.3%) and recall (80.2%) for our approach.

Considering that the relative temporal expressions resolution, there is a great amount of unexplored temporal information in Web pages. Time periods, for instance, express semantic information that is not easily retrieved by conventional search engines. For instance, in the sentence `“Bill Clinton was the American president from 1993 to 2001`”, we have a time period (1993 to 2001) in which a fact (Bill Clinton was the American president) was valid. If a user searches for `“American president 1995”`, the answer should be Bill Clinton. However, a search engine which performs only syntactic searches will not be able to interpret that 1995 is comprised between 1993 and 2001. As a future work, we are extending our current approach in order to use ontologies as repositories for semantic information, rather than storing the extracted facts in a temporal database. User queries, such as “American president 1995”, will be semantically interpreted, so that we will apply our approach to pose queries on on the ontology repository.

# REFERENCES

Arikan, I., Bedathur, S. J., and Berberich, K., 2009. Time will tell: Leveraging temporal expressions in IR. *WSDM*.

Baeza-Yates, R. A. and Ribeiro-Neto, B., 1999. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Boston, MA, USA.

Cherkassky, V. and Yunqian, M, 2004. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17(1):113–126.

Jin, P., Lian, J., Zhao, X., and Wan, S., 2008. Tise: A temporal search engine for web contents. *Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application - Volume 03*. IEEE Computer Society, Washington, DC, USA, pp. 220-224.

Koen, D. B. and Bender, W., 2000. Time frames: temporal augmentation of the news. *IBM Syst. J. vol. 39*, pp. 597-616.

Levine, J. R., Mason, T., and Brown, D., 1992. *lex & yacc (2nd ed.)*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Mani, I. and Wilson, G., 2000. Robust temporal processing of news. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. ACL, Stroudsburg, PA, USA, pp. 69-76.

Maynard, D. and Cunningham, H., 2003. Multilingual adaptations of annie, a reusable information extraction tool. *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics – Volume 2*. EACL '03. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 219-222.

Saquete, E. and Martínez-Barco, P., 2000. Grammar specification for the recognition of temporal expressions.

Sleator, D. D. K. and Temperley, D., 1991. Parsing english with a link grammar. *Third International Workshop on Parsing Technologies*.

Suchanek, F. M., Ifrim, G., and Weikum, G., 2006. Combining linguistic and statistical analysis to extract relations from web documents. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '06. ACM, New York, NY, USA, pp. 712-717.

Verhagen, M. and Pustejovsky, J., 2008. Temporal processing with the tarsqi toolkit. *22nd International Conference on on Computational Linguistics: Demonstration Papers*. COLING '08. ACL, Stroudsburg, PA, USA, pp. 189-192.

XXXXX. This entry was intentionally removed due to the blind review process.

Zhang, Q., Suchanek, F. M., Yue, L., and Weikum, G., 2008. Tob: Timely ontologies for business relations.