Normalization of Temporal Information in Estonian

Margus Treumuth

University of Tartu

Abstract. I present a model for processing temporal information in Estonian natural language. I have built a tool to convert natural-language calendar expressions to semi-formalized semantic representation. I have further used this representation to construct formal SQL constraints, which can be enforced on any relational database. The model supports compound expressions (conjunctive and disjunctive), negation and exception expressions. The system is built to work on Oracle databases and is currently running on Oracle 10g with a PHP web interface.

Keywords: temporal reasoning, temporal reference in discourse

1 Introduction

The goal of my current research is to design and implement the model of calendar expression recognition system for Estonian language that would serve me well as a component for more high level tasks. I have been developing dialogue systems as natural language interfaces to databases and the need for a better temporal normalization has risen from my previous work [1].

One of the most well known applications of calendar expression recognition is Google Calendar "quick add" feature. Users can "quick add" calendar events by typing standard English phrases, such as "Dinner with Michael 7pm tomorrow". Google "quick add" has not been implemented for Estonian.

There is a calendar expression recognition system freely available for Estonian (Datejs - an open-source JavaScript Date Library [2]), yet they have done the straight translation from their English system into Estonian. This approach does not work, as Estonian language is rich in morphology. A simple example of a calendar expression where Datejs would fail is "*esmaspäeval*" ("on Monday"), as this word is not in base form *"esmaspäev"* but has a suffix *-al* meaning that the word is in singular adessive.

Estonian language belongs to the Finnic group of the Finno-Ugric language family. Typologically Estonian is an agglutinating language but more fusional and analytic than the languages belonging to the northern branch of the Finnic languages. The word order is relatively free [3]. A detailed description of the grammatical system of Estonian is available in [4].

Datejs library does not support Estonian agglutinative inflections. Despite of that, Datejs is still a very good example of calendar expression recognition. Taken

all this into account, you'll find quite many references to Datejs throughout this paper. Yet, several other approaches to extracting temporal information are discussed below to provide some comparison.

The system that I built is available online with a web interface and you are welcome to experiment at `http://www.dialoogid.ee/tuvastaja/`.

2 Calendar Expressions

I used a dialogue corpus[1], a text corpus of Estonian[2] and Datejs unit tests to gather calendar expressions that were to be normalized by my system. I also added Estonian national holidays (e.g. *Jõulud* - Christmas Day) and some other less important holidays (e.g. *valentinipäev* - Valentines Day). Here are few examples, to give a basic idea of what expressions need to be handled by temporal normalization:

- *3 aastat tagasi* - 3 years ago
- *14. veebruaril 2004 a* - February 14, 2004
- *kell 17:00* - at 5 PM
- *eelmisel reedel kell 8 õhtul* - last Friday at 8 PM in the evening

Here we see how they could be normalized given that the current date is March 21, 2008:

- 3 aastat tagasi → esmaspäev, 21.03.2005
- 14. veebruaril 2004 a → laupäev, 14.02.2004
- kell 17:00 → reede, 21.03.2008 17:00
- eelmisel reedel kell 8 õhtul→ reede, 14.03.2008 20:00

3 Implementation

I have decomposed the normalization task into following sub-tasks:

1. morphological analysis and creation of n-grams
2. matching rules to calendar expressions
3. calculating the composition of rules based on time granularity measures
4. validating the result against a time line model and providing output in semi-formal notation, SQL notation, ICal format

The database of the normalization model contains rules that are used in the process of normalizing the date expressions. I'll give an overview of the attributes of the rules and then continue by describing the sub-tasks. The normalization model makes use of the power of extended regular expressions. The rules are given as entities with the following five attributes:

[1] `http://lepo.it.da.ut.ee/~koit/Dialoog/EDiC.html`
[2] `http://www.cl.ut.ee/korpused/`

REGEXP - extended regular expression for calendar expression
SEMIFORMAL - corresponding semi-formalized representation of calendar expression
SQL - corresponding SQL representation of calendar expression
ICAL - corresponding ICal representation of calendar expression
GRANULARITY - granularity of the calendar expression (MONTH; DAY; HOUR)

The following example shows how I have defined the rule to normalize the calendar expression *"emadepäev"* - "Mother's Day". We know that the Mother's Day repeats annually on the second Sunday in May. I hereby give an example by describing a rule at first in English and then in Estonian to illustrate the differences of Estonian grammar.

Example 1. Defining a rule to normalize the calendar expression "Mother's Day".

REGEXP: Mother's Day
SEMIFORMAL: DAY= from 08.05 to 14.05 and DAY=Sunday
SQL: <see SQL constraint below this definition>
ICAL: RRULE:FREQ=YEARLY;INTERVAL=1;BYDAY=2SU;BYMONTH=5
GRANULARITY: DAY

The SQL constraint for this rule is defined as follows:

```
to_char(date, 'mm') = 5 and
to_char(date, 'dd') between 8 and 14 and
trim(to_char(date, 'day')) = 'Sunday'
```

The preceding example was translated into English for the sake of readability. The original rule in Estonian has been written as follows:

REGEXP: emadepäev
SEMIFORMAL: PÄEV=08.05 kuni 14.05 ja PÄEV=pühapäev
SQL: <same as in English>
ICAL: <same as in English>
GRANULARITY: <same as in English>

The *REGEXP* attribute contains the base form "emadepäev". There are 14 cases and 2 numbers (singular and plural) in Estonian that are formed by adding suffixes to the base form. Sometimes the base form itself also changes, e.g:

oktoober → *oktoobris* singular inessive (October → in October)
tund → *tunnil* singular adessive (hour → at this hour)

The Estonian word "emadepäev" can have the following 28 possible agglutinative inflections that also need to be matched by the rule:

-a, -ad, -ade, -adega, -adeks, -adel, -adele, -adelt, -adena, -adeni, -ades, -adesse, -adest, -adeta, -aga, -aks, -al, -ale, -alt, -ana, -ani, -as, -asid, -asse, -ast, -ata, -i, -iks, -il, -ile, -ilt, -is, -isse, -ist.

But instead of defining the rule as a long and clumsy regular expression:

emadepäev(a|ad|ade|adega|adeks|adel|adele|adelt
|adena|adeni|ades|adesse|adest|adeta|aga|aks|al|ale
|alt|ana|ani|as|asid|asse|ast|ata|i|iks|il|ile|ilt|is|isse|ist)?

We can define it simply as "emadepäev" and have the morphological analyzer do the rest.

3.1 Morphological Analysis and Creation of N-Grams

The calendar expressions in Estonian contain simple morphology and can be handled manually without fully automated morphological analysis. Yet, it is very inefficient to consider all possible agglutinations manually, I suggest not implementing semantic analysis without proper morphological analyzer [5]. No disambiguation is done in the analysis as the morphological ambiguity in Estonian temporal expressions is very low.

There are currently 59 words that were not handled correctly by the morphological analyzer, e.g *"valentinipäev"*- Valentines Day; *"jüripäev"*- St George's Day. This is a reasonably small amount. These words I handled with specifc regular expressions.

Let's briefly look at morphological analysis with a simple example. Given that the input would be "emadepäevaks" the output of the morphological analyzer would be:

emadepäevaks
ema+de_päev+ks //_S_ sg tr, //

The output of the analyzer shows that the word is a compound noun and that the input was in *singular translative* case. My system is able to parse this output and capture the base form without the -ks suffix.

Date expressions in Estonian language are usually made up of several words. N-grams are created to simplify the parsing process of regular expressions. The rules currently in the system contain at most three words per calendar expression. That is why I created n-grams up to level n=3 (bigrams and trigrams). As most of the rules are given in their base form, the base form n-grams where also created. The n-grams are created at run time just right after the morphological analysis.

3.2 Matching Rules to Calendar Expressions

The third step is finding a rule to match an n-gram, a word or a base form. All words, base forms and n-grams are matched against all rules. Each n-gram, word and base form is constrained to hold a reference to no more than one rule. If a rule is simultaneously referenced by trigram and by a bigram, the reference by the bigram is deleted as it is covered by the trigram. If a rule is referenced by a bigram and by a word, the reference by the word is deleted as it is covered by the bigram (see Example 2). This also means that each n-gram holds a reference to the word or base form that started the n-gram (that is the first word of n-gram). Also, each base form holds a reference to the initial word form.

Example 2. Releasing the matches that were sub-patterns of another match.

1. Consider the input *"järgmisel teisipäeval"*- "next Tuesday" – then the matches would be ***Tuesday*** and ***next Tuesday*** as there is a rule for each of these.
2. We would release the match ***Tuesday*** as it is contained in the bigram ***next Tuesday*** that was also a match.
3. Normalizing "next Tuesday" we would get "*teisipäev, 25.03.2005*" (assuming that current date is March 21, 2008).

3.3 Calculating the Composition of Rules Based on Time Granularity Measures

Conjunctive and disjunctive interpretations are also captured by my system. I have made an empirical judgment that the word "and" in Estonian natural language is mostly used to indicate that it is meant to be handled as disjunctive rather than conjunctive expression. There remains some ambiguity in this approach, yet this assumption seems to serve my purposes quite well. One of the examples of slightly ambiguous compound expression is:

"*sel esmaspäeval ja teisipäeval kell 17*"
"this Monday and Tuesday at 17:00"

Usually this is considered to be a compound disjunctive expression (and so it is handled by my system), giving the output:

(PÄEV=esmaspäev OR PÄEV=teisipäev) AND (KELL=17:00)
(DAY=Monday OR DAY=Tuesday) AND (TIME=17:00)

Yet, depending on wider context, there is a chance that the first part "this Monday" and the second part "Tuesday at 17:00" need to be handled as two separate expressions:

DAY=Monday
DAY=Tuesday AND TIME=17:00

The composition of rules based on time granularity measures is one of the distinct features of my system that has not been so widely exploited by Datejs and Google Calendar.

Above we looked at an example of conjunctive and disjunctive interpretations and saw that temporal units are composed of smaller temporal units. A larger temporal unit is a concatenation of smaller temporal units.

At this point, my system has captured all basic temporal expressions and is ready to compose them into more complex expressions that might represent duration, intervals or recurrence. This is done by taking into account the granularity measures (like hour, day, month). If there are changes in granularity level, the system determines where to add *" AND "* or *" OR "*. A formal framework for the definition and representation of time granularities was proposed by Bettini [6] especially for temporal database applications.

3.4 Validating the Result and Providing Output

Finally the resulting SQL query is validated against a background timeline to see if the conjunctive and disjunctive interpretations that were translated into SQL yield a result. This validation is done for maintenance purposes to capture all misconceptions into the system log.

The output is given by displaying the matched calendar expressions and their formal representations.

4 Experiments and Results

Here are few examples of experiments (in Estonian and in English) that did not pass tests in Datejs but were successfully normalized by my system:

- *valentinipäev* - Valentines Day (this was not even recognized in English in the Google Calendar)
- *igal laupäeval* - every Saturday (recognized as a recurring event)
- *täna ja homme* - today and tomorrow (this was recognized as a disjunctive compound expression as in *3.5* above)

Here are few examples (the test were done in Estonian, yet the examples are given in English) that did not pass tests in my system but were successfully normalized by Datejs:

- *eelmisel aprillil* - last April *(my system does not know the meaning of last - yet)*
- *+5 aastat* - +5 years *(my system does not know the meaning of +)*

I have used some voluntary testers and have received scores from 45% to 55% in recall, and from 75% to 85% in precision.

5 Related Work

Jyrki Niemi and Kimmo Koskenniemi [7] have discussed representing calendar expressions with finite-state transducers, which is basically the same thing that I'm doing - using extended regular expressions to capture the calendar expressions. In their work, they have presented a very good theoretical framework, yet I found no traces of any actual working implementation that was built of this framework.

The automatic time expression labeling for English and Chinese Text by Hacioglu [8] uses a statistical time expression tagger. Yet, they have trained the system on a corpus that has been tagged by a rule-based time expression tagger.

A bachelor thesis was completed by Eveli Saue [9] where a rule-based time expression tagger for Estonian was built (precision of 93% and recall 71%). The tagger does not provide normalized output or any kind of formal representation of the date expression. It just places the recognized time expressions between <TIMEX> tags. In my system I have covered most of the expressions of this tagger was able to tag. I have considered building a tagged corpus and switching to statistical approach, yet the rule-based approach seems easier to control and at the moment I don't have a corpus that would provide a considerable amount of temporal negotiation dialogues.

Saquete et al. [10] present a rule-based approach for the recognition and normalization of temporal expressions and their main point is that this way it was easy to port the system for different languages. They have used a Spanish system to create the same system for English and Italian through the automatic translation of the temporal expressions in rules. I agree with this advantage and can confirm that the porting from Estonian to English would mostly consist of automatically translating the rules. Yet, the translation from English system to Estonian system would not be an easy task as I showed above in the Datejs example.

Berglund [11] has used a common annotation scheme - TimeML (Time Markup Language) and has built a rule-based custom tagger to annotate text with TimeML. He has used the tagger in a system that is used to detect and order time expressions. I have decided to skip the annotation schemes (TimeML and TIMEX) in my output as an intermediate step, as it seemed more convenient for the QA task to produce output in logical expressions that are compatible with SQL.

6 Further Work

The main idea and benefit of current approach is the output of logical expressions that can be used in SQL queries. This approach of temporal normalization is well suited for a smaller domain such as natural language interfaces for relational databases. I will continue by integrating the temporal normalization tool into various natural language interfaces and scheduling applications to see how it would perform. I will extend the model by providing optional constraint relaxation.

The constraint relaxation is implemented in a dialogue system that uses the temporal extraction tool, yet the rules for constraint relaxation are not defined in the temporal extraction tool but in the dialogue system. For example, the user might mention a date to a dialogue system that would result in "not found" response. Then it would be appropriate to relax this date constraint, as in the following dialogue.

<User>: Are there any performances on Saturdays?
<System>: No, yet I found one on this Sunday ...

This was an example of a constraint relaxation where the original date constraint was relaxed by adding one day. This way the users of the system can receive some alternative choices, instead of plain "not found" responses. The constraint relaxation properties can be held in the temporal extraction tool as long as they stay separate from the dialogue domain.

References

1. Treumuth M., Alumäe T., Meister E. A Natural Language Interface to a Theater Information Database. Proceedings of the 5th Slovenian and 1st International Language Technologies Conference 2006 (IS-LTC 2006), pp. 27–30 (2006)
2. Datejs - a JavaScript Date Library, http://www.datejs.com/
3. Kaalep, H.-J. Muischnek, K.: Multi-word verbs in a flective language: the case of Estonian. Proceedings of the EACL workshop on Multi-word expressions in a multilingual context: 11th Conference of the European Chapter of the Association for Computational Linguistics; Trento, Italy; 3 April 2006 Rayson, P. Sharoff, S. Adolphs, S. Association for Computational Linguistics, pp. 57 – 64. (2006)
4. Erelt M. (ed.): Estonian Language. Linguistica Uralica. Supplementary Series, vol. 1. Tallinn: Estonian Academy Publishers. (2003)
5. Kaalep H.-J.: An Estonian Morphological Analyser and the Impact of a Corpus on Its Devel-opment. Computers and the Humanities 31, pp. 115–133. (1997)
6. Bettini, C., Jajodia, S., Wang, X. S.: Time Granularities in Databases, Data Mining, and Temporal Reasoning, Springer-Verlag (2000)
7. Niemi J., Koskenniemi K.: Representing Calendar Expressions with Finite-State Transducers that Bracket Periods of Time on a Hierachical Timeline. In: Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek & Mare Koit (eds.), Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007, pp. 355–362. University of Tartu, Tartu (2007)
8. Hacioglu K., Chen Y., Douglas B.: Automatic Time Expression Labeling for English and Chinese Text. In Proceedings of CICLing-2005, pp 348–359; Springer-Verlag, Lecture Notes in Computer Science, Vol. 3406 (2005)
9. Saue E.: Automatic Extraction of Estonian Temporal Expressions, http://math.ut.ee/~sints/bak/ (2007)
10. Saquete E., Marinez-Barco P., Munoz R.: Multilingual Extension of a Temporal Expression Normalizer using Annotated Corpora, Cross-Language Knowledge Induction Workshop (2006)
11. Berglund A.: Extracting Temporal Information and Ordering Events for Swedish. Master's thesis report (2004)