Implementing “Quick add” feature in calendar application using Flex and Natural language processing

Anchal Agarwal[1], Ela Rathee[2], Narina Thakur[3]

Department of Computer Science
BVCOE, Affiliated to GGSIPU
New Delhi, India

Abstract

With the advent of Natural language processing communications between human and computers become much at ease. Natural Language Processing (NLP) is automated process for analyzing text based on set of theories and technologies. This paper proposes natural language processing for calendar application. This NLP application lets a user add an event in a calendar without filling manual form for adding a new event. The proposed system accepts a phrase in natural language, interpret it, derive logical meaning and add a new event in the calendar application. The input phrase is processed using pattern matching technique of natural language processing to generate tokens which are formal representations that are easier to be manipulated by the application. Flex is explored and used to automate the lexical analysis process.

Keywords—Quick add; Natural language processing; pattern matching; Flex; Lexical analysis.

I. INTRODUCTION

Nowadays, Calendar and Scheduling applications are being greatly used to schedule events, todos, reminders, journals, etc. by all sorts of people viz students, professionals. The most common use of these applications is scheduling of events which may also act as reminders for the users.

Event addition in any calendar application requires the user to fill out a form for a new event, giving a number of details like description, date, start time, end time, location, people involved, summary, etc. This enables the users to add events with detailed specifications. But sometimes, all what a user intends to do is to add an event with only a few attributes say description, date and time. In such cases, filling out the entire forms for adding a new event becomes a long and time consuming process. Also, sometimes the users want to add events quickly without spending too much time in filling out fields of a new event's form. Thus, there arises the need for a plugin or an application that will help the users to carry out event addition quickly.

Quick add feature serves this purpose and allows a user to quickly add events in a calendar without having to fill the form for new event. The user can simply enter a phrase in natural language like “meeting with HOD in conference room at 3 pm on 22-04-2013”. This application processes the phrase, analyses and interprets it and derives the logical meaning by tokenization
on the input and generates 3 tokens: 'event', 'time' and 'date'. Having generated the tokens, the appropriate event is added to the calendar with description or summary as the 'event' on the date specified by 'date' at time specified by 'time'. This feature is already present in iCal (Apple's personal calendar application) and Google Calendar, none of which is open-sourced.

This calendar application uses KOrganizer which is an open-sourced calendar and scheduling component of the Kontact suite developed by KDE [1]. KOrganizer is intended to be a complete program for organizing appointments, contacts, projects, etc. KOrganizer intuitively supports information interchange with other calendar applications, via vCalendar personal data interchange file format to ease the transfer from other PIMs to KOrganizer. [2]Section II discusses natural language processing and its techniques, Section III explains in detail the technique employed by us i.e. Pattern matching followed by discuss in detail the design, the tool used, the regular expressions formulated, and the acceptable formats of the input phrase and implementation in Section-IV.

II. NATURAL LANGUAGE PROCESSING

Natural language processing is an upcoming active area in the research field. Natural languages are the languages which have naturally evolved and used by human beings for communication purposes. Natural Language Processing or NLP (also called Computational Linguistics) [3] is the scientific study of languages from computational perspective. Natural Language processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and human (natural) languages. [4]

NLP was formerly named as Natural language understanding (NLU) in Artificial Intelligence. NLU systems translate samples of human language into more formal representations that are easier for computer programs to manipulate. There are several NLP techniques available viz pattern matching, syntactically driven parsing, semantic grammars, etc. [5]

NLP system processing steps are:

- Paraphrase an input text
- Decode the text into another language
- Elucidate the contents of the text
- Draw inferences from the text

The NLP technique used to process the input in this case is pattern matching or lexical analysis. [6] The core gist of pattern matching is to process the input and derive a logical meaning by matching patterns of words against the input occurrence. Regular expressions are used to specify these patterns.

A. LEXICAL ANALYSIS OF THE INPUT PHRASE

The stream of characters making up the input in natural language is read from left to right and
grouped into tokens. This process is called lexical analysis or tokenization. Tokens are sequences of characters with a collective meaning. There are mainly three tokens in this case: event, date and time. From the area of compilers, we get a number of tools to perform lexical analysis.

B. FLEX

Flex, the fast lexical analyzer, has been used. It takes a specification file called the lex file as input and generates an analyzer, normally called as lex.yy.c.

The lex files contains rules for patterns expressed as regular expressions which provide a powerful means to "match" (specify and recognize) strings, such as specific characters, words, or patterns of characters.[7]

Actions are associated with each pattern. Actions are C source fragments. If it is compound, or takes more than one line, it is enclosed within braces ("{ ""). Occurrence of event, date and time patterns results in the display of the matched substring. This lex file is fed as an input to the flex tool. The output is a lexical analyzer (lex.yy.c). The input string is an input to this lexical analyzer which produces event, date and time tokens.

C. REGULAR EXPRESSIONS

Regular expressions are specially encoded text strings used as patterns for matching sets of strings. They are all about matching and finding simple as well as complex patterns in the given text. The pattern matching can be done using string literals, digits, letters and/or characters of any kind. [8]

We have taken into consideration set of the most probable input formats and designed the regular expressions accordingly.

The regular patterns used are:

Prep (on | at | " "on|" "at|" "on|" "at|" "on|" "at" )
word ([a-zA-Z]+)[ ]?
event {word}+[ ]at[ ]{word}+{word}+
date1 [1-3]?[0-9]"/"[1]?[0-9]"/"[2][0-1][0-9][0-9][1-3]?[0-9]"- "[1]?[0-9]"- "[2][0-1][0-9][0-9][1-3]?[0-9]
date2 [0-3]?[0-9][ ]jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec
date {date1}{{date2}
time ([1]?[0-9][ ][a-p]m)|([1]?[0-9]":[0-5][0-9][0-9]|([1-2]?[0-9][0-9])|([1-2]?[0-9]":"[0-5][0-9][0-9])}

D. FINITE AUTOMATAS

Finite Automata is an input processing machines where, there is an input alphabet and each input string causes the machine to output either 'yes' or 'no'. Those strings that cause the machine to
output 'yes' are said to be accepted by the finite automata, and those strings that cause it to output 'no' are said to be rejected by Finite automata. Every regular expression is Finite Automata recognizable. [9]

The finite automatas for regular expressions listed in section IV (B) are as follows:

Fig 1. Automata for "prep" expression

Fig 2. Automata for "word" expression

Fig 3. Automata for "event" expression

Fig 4. Automata for "date1" expression

Fig 5. Automata for "date2" expression

Fig 6. Automata for "date" expression
E. ACCEPTABLE FORMATS

The input formats acceptable are:

- `<event>` at `<venue>` at `<time>` on `<date>`
- `<event>` at `<time>` on `<date>`
- `<event>` at `<venue>` on `<date>` at `<time>`
- `<event>` at `<venue>` on `<date>` at `<time>`

where the `<date>` can take following forms:
- 25 dec
- 25-12-2012
- 25/12/2012

and `<time>` can take following forms:
- 13:00
- 1 pm
- 1:00 pm
- The `<venue>` is treated as a part of `<event>`.

EVENT ADDITION

Event adder is the program which adds the event onto the calendar. It reads the output of flex which is nothing but three tokens viz event, date and time. The calendars in KOrganizer are saved as `.ics` files.
It is a computer file format which permits the Internet users to send meeting requests and tasks to other people. iCalendar is used by various products like Apple's iCal, Google Calendar, KOrganizer, etc.

This format is independent of the transport protocol. The element present at the highest hierarchy in iCalendar is the Calendaring and Scheduling Core Object. Its body consists of calendar properties and one or more calendar components like an event, a to-do list, a journal entry, time zone information or an alarm. [10]

**VEVENT** describes an event which has a scheduled amount of time on a calendar. The format for a **VEVENT** is given as follows:

```
BEGIN:VEVENT
DTSTART:
DTEND:
SUMMARY:
LOCATION:
DESCRIPTION:
PRIORITY:
END:VEVENT
```

The event adder opens the `.ics` file of the default calendar. It then creates a code for an event in the above specified format using the output of the flex. Default values are used for the attributes whose values are not specified by the input string. [11] [12]. The snippet thus created is inserted in the `.ics` file at appropriate place by the event adder. KOrganizer continuously monitors this `.ics` file and thus the event is popped onto the calendar as soon as the addition is made.

### III. IMPLEMENTATION

To summarize, the following course of actions take place every time the quick add feature is used:
The 'Quick add' feature is useful for creating simple events in lesser time and in an efficient way. Since we have considered the most probable set of inputs, this feature works for only a limited set of input formats. Quick add feature adds events to the calendars of iCalendar format which is a standard Internet calendar format used by a number of calendar applications available in the market. Thus, this feature can be ported to other open sourced applications also which can deal with .ics format of the calendar.

IV. CONCLUSION & FUTURE SCOPE

This paper discusses Natural language processing and lexical analysis. The efficiency of the proposed application can be enhanced by accommodating/ training the system with recurring events. Recurring events are those which are repeated over a time period like weekly, daily or monthly events. The application can be modified to recognize, accept and process recurring events using the keywords like 'daily', 'weekly', 'monthly to match the patterns. Also, the number of acceptable formats can be increased allowing several other patterns for the input string to be processed by the application. The event addition, as of now, is being carried out using Command Line Interface. A Graphical User Interface may be provided to the user like a pop-up menu where the user can enter the input string in a textbox.

REFERENCES


[3] Mehnaz khan, Dr. Mehraj-ud-Din Dar, Dr. S.M.K. Quadri

[4] Tim Flhin , Aravind K. Joshi, Bonlle Lynn Webber,
   “Natural language interactions with Artificial experts”, MS-CIS- 86-16, LINC LAB - 08

[5] Mehnaz khan, Dr. Mehraj-ud-Din Dar, Dr. S.M.K. Quadri

[6] Tim Flhin , Aravind K. Joshi, Bonlle Lynn Webber,
   “Natural language interactions with Artificial experts”, MS-CIS- 86-16, LINC LAB - 08


