Term Project

Junye Zhu N11235693

Assignment: Extract time expression from news articles and normalize them into standard time expressions.

The assignment contains two parts: “Time extraction” and “Time normalization”.

**Time Extraction:**

There are so many time expressions, format of which varies a lot. The first thing we need to do is simplify the model. After all useless adjectives and prepositions removed, following general formats can be easily summarized:

Keywords General Formats

On On Friday/Christmas/Sep. 1st/1st of Sep./2rd Friday of Sep.

In In 2012/In April 2012/3 month

At At Friday night/night of Friday/night of Christmas

This/That/Last/Next Last Friday/Christmas/night/April/week

Tomorrow/Yesterday/Today

Month or number two days ago/April 17 2012/

We notice that although formats of time expressions varies, the words used by the time expressions are almost fixed, making dictionary matching a very good candidate solution. We created following several word lists:

List Content

MONTHS.list all months and their variants

WEEKDAYS.list all weekdays and their variants

HOLODAYS.list all holidays and their variants

DURINGDAY.list night, evening and so on.

SINGLEDAY.list today, yesterday, tomorrow

ORDERS.list this, that, last, next

GENERALTIMES.list day, month, year, decade and so on.

NUMBERS: numbers and its variants

Things are much simpler when all the preparation work done. We split the article to separated sentences at first since it is obvious that matching formats in a sentence is much simpler the whole article. The problem now is matching all the general formats in a given sentence.

C++ was used to solve the problem with the following steps:

1. Tokenization. Tokenize the give sentence and match the first word of each format with the first token to determine the general format.

2. Match the rest of the sentence with the format determined above. In order to improve its scalability, we choose to write a different matching functions for each format so we can easily add new functions if we get new formats in the future. Currently we have “processKeywordOn()”, “processKeywordIn()”, “processKeywordAt()”, “processKeywordSingleDay”, “processKeywordNumber()”, “processKeywordMonth()” and “processKeywordOrder()”. In each function, we compare the token with the expected word or check if the token in list of expected words, and then do it again util the match completes.

**Time Normalization:**

Time normalization is much simpler than time extraction since it doesn’t has so many different formats, but there are also some problem should be noticed.

First, words like “April” and “1st” in expression “April 1st 2012” should be translated to numbers or indexes such as “01042012”, so map files from words to numbers are necessary here. Map files use by the program are as follows:

Maps Content

MONTHS.map english words of months to its index

WEEKDAYS.map english words of weekdays to its index

HOLIDAYS.map holidays to its standard time format. eg. Christmas to 2512

NUMBERS.map ordinal numbers and words number like “one” to digit number

Second, there are two kinds of time format we need to normalize, the absolute time and the relative time. The absolute time is like “on April 17 2001”; it has nothing to do with the input and easy to handle. However, there are also relative time such as “2 days later” or “this Friday”, in the case of which we need to count the “diff” between the input time and result to get the correct value. We have different functions take month, year, day, weekday or holiday as parameters and calculate the new date according to the parameter and current date. Currently, we have functions “countWeekDay()”, “countHoliday()”, “countMonth()”, “countMonthAndDay()”, “coutWeekAndMonth()” and “tellGeneralTimes()”. Each of the function will set the new date according to related rules, either by counting the result date directly or count the “diff days” and calculate the result date.

**Achievement:**

We take a news from New York Times and do a experiment using the program. The results are pretty good, with most of time expressions are extracted. Here are the results:

Experiment Result( with input date “02052014”):

Got: In March (01032014)

Got: In 2013 (01012013)

Got: In July 2012 (01072012)

Got: APRIL 1 (17002014)

Got: In 2012 (01012012)

Got: In 10 years (02052024)

Got: this year (02052014)

Got: April 13 (13042014)

Got: In 2013 (01012013)

Got: On Wednesday (30042014)

Got: last month (02042014)

Got: In March (01032014)

Got: On March 28 (28032014)

Got: 10 days later (12052014)

Got: In 2012 (01012012)

Got: In 2012 (01012012)

Got: In 2009 (01012009)

Got: four years later (02052018)

Got: In July (01072014)

Got: In January 2013 (01012013)

Got: last year (02052013)

Got: In March (01032014)

Got: last year (02052013)

Got: June 16 (16062014)

Got: last month (02042014)

Got: May 4 (04052014)

In this article, I got three mistakes.

Like ‘Through April 13, officers seized roughly 1,700 glassine bags of heroin, up from about 1,200 bags over the same period in 2013. ’

Got: April 13(13042014)

Got: In 2013(01012013)

**Run:**

To run my program, just run ‘./run.sh’ in iterm. The input file is the test article. And the input date is written in TimeExtraction.cpp.