IFT6285 Traitement Automatique des Langues Naturelles
Professor: Philippe Langlais
Student: Gustavo Alonso Patino Ramirez

Homework 2

1. In Appendix we can see the python code use for generating the table *table freq.txt*, the format of the table is three columns representing the 1000 most frequent words, its rang and how many times it appears in the corpus, before generating the table the words were filtered to map only alphabetic words in lowercase. The lecture of the input corpus was done with the package pandas and the table was generated using a python table as can be see in the function *build frequency*

2. In the Appendix we can see the code for normalization, the normalization was done in several steps:

   - Management of punctuation mark{ ! or ...} : In the function *clean text* we check these punctuation marks and were replaced by {STRONG!, DOTS }
   - Deformed words: Using the package *collections* inside the function *final text* we identified words with more than four repeated characters, avoiding words as caroliiine or goood. In the case of words as heheh or hahah we did a replacement by the word RIRE.
   - Filters were made for emojis in the function *icons*. Specifically the icons: *set*{ :D, :) , :-), :-], :->, 8-), :), :-}, :), :], :3, :>, 8), :}, :c) } were replaced by the word SMILY while emojis as: set { :(, :-c, :-<, :<, :[, >:[, :{ } were replaced by the word SAD.
   - Normalization of numerical entities: Objects as 10:30am, 9:35pm, 5pm, 8am were normalized by the word TIME in the function *clean text*

   The input file *train.csv* was imported and all the corpus was normalized using the rules described above and written in the file *normalise.csv*, respecting the original order and format, that is, the document is structured as: "text," label. The total number of replacements in the train corpus of 1000 text was 4596
   As an example, we show the normalization of three different texts:

- Text: caroliiiiiine... she's mighty fiiiine.... omg heheh!!! none :(of those ceiling. tiles are straight hahah!!!, so 10:30am!
  - Normalization: caroline DOTS she mighty fine DOTS omg RIRE STRONG! none SAD of those ceiling tiles are straight RIRE STRONG! so TIME STRONG!
- Text: ooooh!! i found a cool new theme for firefox called noia. it is all snazzy and shinny, you can get your own urllink here. urllink they are gooooooood.
  - Normalization: oh STRONG! i found a cool new theme for firefox called noia it is all snazzy and shinny you can get your own urllink here urllink they are good
- Text: hahaha, it was soooo ironic that we watched that tv show before doing this!!! ohio!!! so this is emily and sara all prepped for dyeing!!! urllink urllink dsc00286 originally uploaded by urllink vivaldica. urllink me and bike.... dont i look gooood! urllink
  - Normalization: RIRE it was so ironic that we watched that tv show before doing this STRONG! ohio STRONG! so this is emily and sara all prepped for dyeing STRONG! urllink urllink originally uploaded by urllink vivaldica urllink me and bike DOTS dont i look good STRONG! urllink

At follows you can see some pieces of the code, a complete version of the code was not provided because the document exceeds the size allowed. A complete version can be sent if required.

APPENDIX

```
def build_frequency(data, number):
    dic_mf={}
    len=data.shape[0]
    for i in range(len):
          print(i, "of", len)
          text=data[i,0]
          clean_words=token_words(text)
          for word in clean_words:
              if(word in dic_mf):
                  dic_mf[word]+=1
              else: dic_mf[word]=1

    words_sorted=sorted(dic_mf.items(), key=lambda x: x[1], reverse=True)
    freq_words={}

    wordsF=words_sorted[0:number]
    count=0
    for tup in wordsF:
          count+=1
          w,f=tup
          freq_words[w]=[count,f]
   return freq_words
```

Code for normalization

```
import pandas as pd
import nltk
import re, string, unicodedata
from collections import OrderedDict
import collections
import csv

def clean_text(text):
   #split by several dots
   sr = re.split("\.\.+", text)

   string_n=""
   for i in range(len(sr)):
       token=sr[i]
       if(i<len(sr)-1):
           string_n+=token+" DOTS"+" "
       else:
           string_n+=token

   #split by several!
   sr = re.split("\!+", string_n)

   string_n=""
   for i in range(len(sr)):
      token=sr[i]
      if(i<len(sr)-1):
         string_n+=token+" STRONG"+" "
      else:
         string_n+=token

   #split by time format hour:min!
   pattern="\d:\d"
   sr = re.split(pattern, string_n)

   string_n=""
   for i in range(len(sr)):
      token=sr[i]
      if(i<len(sr)-1):
         string_n+=token+" TIME"+" "
      else:
         string_n+=token


   #split by time format hour pm!
   pattern="\dpm"
   sr = re.split(pattern, string_n)

   string_n=""
   for i in range(len(sr)):
      token=sr[i]
```

```
        if(i<len(sr)-1):
            string_n+=token+" TIME"+" "
        else:
            string_n+=token

    #split by time format hour pm!
    pattern="\dam"
    sr = re.split(pattern, string_n)

    string_n=""
    for i in range(len(sr)):
        token=sr[i]
        if(i<len(sr)-1):
            string_n+=token+" TIME"+" "
        else:
            string_n+=token


    return string_n


def icons(text):
  icons=[":D",":\)", ":-\)",':-]',':->','8-\)', ':\)',':-}', ':\)',':]',
         ':3', ':>', '8\)', ':}', ':c\)',':^\)']

  icons2= [":âĂŚ\(",":\(", ":-c",":-<",":<",":\[", " >:\[",  ":{"]


  string_n=text
  for icon in icons:
     sr = re.split(icon, string_n)
     #icon found
     if(len(sr)>1):
          string_n=""
          for i in range(len(sr)):
             token=sr[i]
             if(i<len(sr)-1):
                string_n+=token+" SMILY"+" "
             else:
                string_n+=token

  for icon in icons2:
     sr = re.split(icon, string_n)
     #icon found
     if(len(sr)>1):
          string_n=""
          for i in range(len(sr)):
             token=sr[i]
             if(i<len(sr)-1):
                string_n+=token+" SAD"+" "
             else:
                string_n+=token
 return string_n


def final_text(text):
    ft=""
    tokens=nltk.word_tokenize(text)


    for token in tokens:
          results = collections.Counter(token)
          max_value = max(results.values())
          if(max_value>=4):
             result = "".join(OrderedDict.fromkeys(token))
             if(result.isalpha()):
                  if(result=="STRONG"): result="STRONG!"
                  elif(result=="he"): result="RIRE"
                  elif(result=="ha"): result="RIRE"
                  ft+=result+" "
          else:
             result=token
             if(result.isalpha()):
                  if(result=="STRONG"): result="STRONG!"
                  ft+=result+" "
    return ft
```