Homework 1 for MSIA 414 Text Analytics

By Brennan Antone

Problem 1

I focused primarily on investigating text pre-processing using the stanfordnlp and the nltk libraries in python. Overall these libraries were pretty comparible - since I was working with a small text corpus, I noticed no practical differences in runtime. However, I did think that the nltk package was easier to use, perhaps because I liked the documentation on their website better.

One thing in particular that I was interested in was Chinese text, since for my personal research I am working with a dataset of messages that were shared with me by a Chinese company. The text contains messages on an enterprise social media platform (similar to Slack or Microsoft teams) that the company uses. The dataset has a little over 64,000 messages.

Because of this, I spent a lot of time focusing on tokenization, in particular to see if I could find a pretrained model to perform word segmentation on the Chinese text (adding spacing to group characters in reasonable units for NLP tasks). After a lot of investigation, I concluded that the Stanford Word Segmenter for Chinese would be good for my purposes. Though it was pre-trained on more formal text, which may present problem with more collequial language potentially being in my dataset, I still wanted to look at using a pretrained parser since my dataset is not massive.

Based on some googling, Stanford NLP seemed to have good functionality for this, and seemed easy enough to use. I was able to perform word segmentation using the Stanford Segmenter, and it was, of course, incredibly quick because I was using a pre-trained model. The Stanford Natural Language Processing Group website actually directed me to download an executable that did this processing outside of python (inputting commands through terminal). This was not a particularly satisfying way to do this, since it could be potentially limiting to have to do a lot of pre-processing out of python.

Fortunately, after some more searching, I realized that the nltk package actually had a nltk.tokenize.stanford_segmenter function that handles this! Ironically enough, this was actually the version of the stanford word segmenter that I found the easiest to use.

Problem 2

Load a portion of 20 newsgroups text corpus

In [2]:
```
import sklearn.datasets
newsgroups_train = sklearn.datasets.fetch_20newsgroups(subset='train')
```

Email RegEx

I choose to use parts of the recommended 20 Newsgroups corpus for this exercise. For email, I started by writing a very broad code and then adding a couple additional constraints to narrow down what forms of text I was getting.

In [27]:
```
# import re

text = """ If you need, to email wba@AOL.com to call 1-800-935-9935. Fro
m the  US, 1-405-272-9935. debit Debit """

# I went through several versions of the email filter, iteratively addin
g features
test1 = re.findall('\S+@\S+', text) # char @ char model
                                    # Want to switch to a compilable mode
l
test2 = re.compile(r"\S+@\S+[a-z]", flags=re.I) # Converts to lowercase,
look for alphanumeric after email
test3 = re.compile(r"[a-z0-9]+@[a-z0-9]+[a-z0-9]", flags=re.I)
# Fixing issue where "<wba@aol.com" would include "<"
# Attempt 3 was too strict, needed to broaden it a bit
test4 = re.compile(r"\b[a-z0-9.%]+@[a-z0-9]+[a-z0-9.%]+", flags=re.I) #
 More flexible with periods, etc.

print(test1)
print(test2.findall(text))
print(test3.findall(text))
print(test4.findall(text))
```

```
['wba@AOL.com']
['wba@AOL.com']
['wba@AOL']
['wba@AOL.com']
```

In [28]:

```
# Test on proportions of the 20 Newsgroups data
print("Example:")
print(newsgroups_train.data[1])
print("Attempt 2:")
print(test2.findall(newsgroups_train.data[1]))
print("Attempt 3:")
print(test3.findall(newsgroups_train.data[1]))
print("Attempt 4:")
print(test4.findall(newsgroups_train.data[1]))

print("")
print("")
print("Apply the final regex to the first 50 texts:")
for x in range(50):
    print(test4.findall(newsgroups_train.data[x]))
```

```
Example:
From: guykuo@carson.u.washington.edu (Guy Kuo)
Subject: SI Clock Poll - Final Call
Summary: Final call for SI clock reports
Keywords: SI,acceleration,clock,upgrade
Article-I.D.: shelley.1qvfo9INNc3s
Organization: University of Washington
Lines: 11
NNTP-Posting-Host: carson.u.washington.edu

A fair number of brave souls who upgraded their SI clock oscillator hav
e
shared their experiences for this poll. Please send a brief message det
ailing
your experiences with the procedure. Top speed attained, CPU rated spee
d,
add on cards and adapters, heat sinks, hour of usage per day, floppy di
sk
functionality with 800 and 1.4 m floppies are especially requested.

I will be summarizing in the next two days, so please add to the networ
k
knowledge base if you have done the clock upgrade and haven't answered
this
poll. Thanks.

Guy Kuo <guykuo@u.washington.edu>

Attempt 2:
['guykuo@carson.u.washington.edu', '<guykuo@u.washington.edu']
Attempt 3:
['guykuo@carson']
Attempt 4:
['guykuo@carson.u.washington.edu', 'guykuo@u.washington.edu']


Apply the final regex to the first 50 texts:
['lerxst@wam.umd.edu']
['guykuo@carson.u.washington.edu', 'guykuo@u.washington.edu']
['twillis@ec.ecn.purdue.edu', 'twillis@ecn.purdue.edu']
['jgreen@amber', 'rob@rjck.UUCP', 'abraxis@iastate.edu', 'abraxis.73434
0159@class1.iastate.edu', 'jgreen@csd.harris.com']
['jcm@head', 'C5owCB.n3p@world.std.com', 'tombaker@world.std.com', 'C5J
Lwx.4H9.1@cs.cmu.edu', 'ETRAT@ttacs1.ttu.edu']
['dfo@vttoulu.tko.vtt.fi', '4t@transfer.stratus.com', 'cdt@sw.stratus.c
om', '1993Apr20.083057.16899@ousrvr.oulu.fi', 'dfo@vttoulu.tko.vtt.fi',
'4j3@transfer.stratus.com', 'cdt@sw.stratus.com', 'C5n3GI.F8F@ulowell.u
lowell.edu', 'jrutledg@cs.ulowell.edu', 'cdt@rocket.sw.stratus.com', 'c
dt@vos.stratus.com', 'douglas.foxvog@vtt.fi']
['bmdelane@quads.uchicago.edu', 'bmdelane@midway.uchicago.edu']
['bgrubb@dante.nmsu.edu', 'DXB132@psuvm.psu.edu', '1qlbrlINN7rk@dns1.NM
SU.Edu', 'bgrubb@dante.nmsu.edu']
['holmes7000@iscsvax.uni.edu']
['kerr@ux1.cso.uiuc.edu', 'jap10@po.CWRU.Edu', 'stankerr@uiuc.edu']
['irwin@cmptrc.lonestar.org', 'irwin@cmptrc.lonestar.org']
['david@terminus.ericsson.se', 'david@terminus.ericsson.se', '17570@fre
enet.carleton.ca', 'ad354@Freenet.carleton.ca', 'david@terminus.ericsso
```

```
n.se']
['rodc@fc.hp.com', 'rodc@fc.hp.com']
['dbm0000@tm0006.lerc.nasa.gov', '1993Apr23.184732.1105@aio.jsc.nasa.go
v', 'kjenks@gothamcity.jsc.nasa.gov']
['jllee@acsu.buffalo.edu']
['mathew@mantis.co.uk', 'kmr4@po.CWRU.edu']
['ab@nova.cc.purdue.edu', 'prestonm.735400848@cs.man.ac.uk', 'prestonm@
cs.man.ac.uk']
['CPKJP@vm.cc.latech.edu', 'dans@jdc.gss.mot.com', 'Callison@uokmax.ec
n.uoknor.edu', 'Callison@aardvark.ucs.uoknor.edu', 'goldberg@oasys.dt.n
avy.mil', 'stack@translab.its.uci.edu', 'anisetti@informix.com']
['ritley@uimrl7.mrl.uiuc.edu', 'ritley@uiucmrl.bitnet']
['abarden@tybse1.uucp', 'abarden@afseo.eglin.af.mil']
['keith@cco.caltech.edu', 'livesey@solntze.wpd.sgi.com']
['leunggm@odin.control.utoronto.ca', '1993Apr20.151818.4319@samba.oit.u
nc.edu', 'Scott.Marks@launchpad.unc.edu']
['rpwhite@cs.nps.navy.mil']
['csyphers@uafhp..uark.edu', 'ssa@unity.ncsu.edu', 'ssa@unity.ncsu.ed
u']
['nodine@lcs.mit.edu']
['kph2q@onyx.cs.Virginia.EDU', 'kph2q@onyx.cs.Virginia.EDU', 'kph2q@vir
ginia.edu']
['nagle@netcom.com', 'd7q@sequoia.ccsd.uts.EDU.AU', 'swalker@uts.EDU.A
U']
['r4938585@joplin.biosci.arizona.edu']
['jonh@david.wheaton.edu', 'Apr.5.23.31.36.1993.23919@athos.rutgers.ed
u', 'by028@cleveland.freenet.edu', 'jhayward@imsa.edu']
['jimf@centerline.com', 'tcorkum@bnr.ca', 'jimf@centerline.com']
['mrh@iastate.edu', 'C5t7qG.9IJ@rice.edu', 'xray@is.rice.edu']
['pchurch@swell.actrix.gen.nz', 'pchurch@swell.actrix.gen.nz']
['xandor@unixg.ubc.ca']
['ayr1@cunixa.cc.columbia.edu', 'ayr1@cunixa.cc.columbia.edu', '2528@sp
am.maths.adelaide.edu.au', 'jaskew@spam.maths.adelaide.edu.au', '1993Ap
r13.002118.24102@das.harvard.edu', 'adam@endor.uucp', '1993Apr12.18403
4.1370@bnr.ca', 'zbib@bnr.ca', 'jaskew@spam.maths.adelaide.edu']
['joec@hilbert.cyprs.rain.com', 'Apr.9.08.39.25.1993.15639@romulus.rutg
ers.edu', 'kaldis@romulus.rutgers.edu', 'kaldis@remus.rutgers.edu']
['dchhabra@stpl.ists.ca', '120666@netnews.upenn.edu', 'kkeller@mail.sa
s.upenn.edu']
['static@iat.holonet.net']
['ebrandt@jarthur.claremont.edu', 'an3@news.intercon.com', 'amanda@inte
rcon.com', 'ebrandt@jarthur.claremont.edu']
['behanna@syl.nj.nec.com', '140493155620@b329', 'tcora@pica.army.mil',
'1993Apr14.125209.21247@walter.bellcore.com', 'fist@iscp.bellcore.com',
'behanna@syl.nj.nec.com']
['bressler@iftccu.ca.boeing.com', 'vincent@cad.gatech.edu']
['1993Apr17.213553.2181@organpipe.uug.arizona.edu', 'krueger@helium.ga
s.uug.arizona.edu']
['root@ncube.com', 'root@ncube.com', 'zod@ncube.com']
['ab245@cleveland.Freenet.Edu']
['paul@csd4.csd.uwm.edu', 'paul@csd4.csd.uwm.edu']
['cmeyer@bloch.Stanford.EDU', 'mike@cunixf.cc.columbia.edu']
['93105.152944BR4416A@auvm.american.edu']
['vng@iscs.nus.sg', 'VNG@ISCS.NUS.SG']
['speedy@engr.latech.edu', 'csundh30.735325668@ursa', 'csundh30@ursa.ca
lvin.edu']
['tg@cs.toronto.edu', 'tg@utstat.toronto.edu']
```

```
['18084TM@msu.edu', 'isu@VACATION.VENARI.CS.CMU.EDU', '18084tm@ibm.cl.msu.edu']
```

Date RegEx

For date, I also started broad, but this was trickier. I decided that, for a date, having some sort of numeric component (ie. year, calendar day, month) was required, but these components could actually be in any sort of order.

In [23]:

```
import re

text = """ If you need, to email wba@AOL.com to call 1-800-935-9935. Fro
m the  US, 1-405-272-9935. debit Debit """

# I want to build a filter to identify dates
# I will assume all dates have a month/year or a
#date1 = re.compile(r"0[123456789]|10|11|12)([/])(([1][9][0-9][0-9])|
([2][0-9][0-9][0-9]+", flags=re.I) # More flexible with periods, etc.
date1 = re.compile(r"([0-9]|10|11|12)[/]") #This one wasn't strict enoug
h
# Google indicates that I can use \d for represeting digits
date2 = re.compile(r"(\d+/\d+/\d+)")


print("Apply the final regex to the first 50 texts:")
for x in range(300):
    print(date2.findall(newsgroups_train.data[x]))
```

```
Apply the final regex to the first 50 texts:
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
['331/3411/158288']
[]
```

[]
[]
[]
['93/04/01', '2/20/93', '10/06/97', '04/21/98', '06/20/99', '08/16/99',
'12/30/00', '06/25/04', '01/09/05', '01/30/05', '06/25/08', '7/22/92',
'10/18/89', '02/09/90', '12/08/90', '05/01/91', '10/29/91', '12/08/92',
'08/28/93', '07/02/95', '07/09/95', '12/07/95', '07/18/96', '09/28/96',
'12/12/96', '01/23/97', '02/28/97', '04/22/97', '05/31/97', '10/05/97',
'1/24/90', '9/25/92', '8/24/93', '8/10/92']
[]
[]
[]
[]
[]
[]
[]
['3/28/93']
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]

[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
['93/04/01']
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]

[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
['4/15/93']
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]

[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]

[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]

Problem 3

Machine learning is an area of research that attempts to use algorithms, trained on data, in order to make informed predictions of decisions. Often, these decisions may involve text: evaluating sentiment, identifying dates, selecting certain types of documents, etc. To accomplish this, a first step that some models attempt to use is parsing a written sentence to identify parts of speech. (And sometimes this may be a goal for models in and of itself.) These parts of speech can then be used in subsequent analysis. By "parsing" a sentence, we mean coming up with a strict mathematical representation of what the text looks like. Researchers have previously come up with one such representation, called a Penn tree-bank style parse tree, that decomposes a sentence by assigning a set of categories to words and a set of relationships between words in categories, within a nested tree-like structure (Marcus, Santorini, & Marcinkiewicz, 1993). Then, a standard goal in machine learning is develop an algorithm that can learn, based on a set of text used for training, how to automatically develop these Penn tree-bank style parsers for new text.

The paper "A Maximum-Entry Inspired Parser" (Charniak 2000) presents a new method for automatically creating these Penn tree-bank style parsers. They take a statistical approach in developing a mathematical model for representing the likelihood of different types of valid Penn tree-banks and identifying the Penn tree-bank that is most likely to be correct for a given sentence. The joint probability distribution from all parts of a tree may be expressed in the form of a top-down nested expression: First, considering every part of the sentence (noun phrase, verb phrase, etc.), then every word label for that part of sentence, every part of sentence branching off of that, etc. The probability of the whole parse is then proportional to the probability of each of the parts given the others. Probabilities are found by considering other covariate features, alongside using a Markov-based model for representing word co-occurrences. Rather than apply this type of probabilistic model to consider all possible parses, since the number to consider for every sentence would be enormous, the authors generate a set of "candidate" parses, using an existing approach known as a chart parser (Caraballo, & Charniak, 1998), and compare these parses to select the best possible solution. To demonstrate the merit of this new algorithm, Charniak compares his new method to a prior method used in the field at the time, and demonstrates that on various text corpuses, his new method has increased predictive performance (in terms of model precision and recall).

Works Cited

Caraballo, S. A., & Charniak, E. (1998). New figures of merit for best-first probabilistic chart parsing. Computational Linguistics, 24(2), 275-298.

Charniak, E. (2000, April). A maximum-entropy-inspired parser. In Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (pp. 132- 139). Association for Computational Linguistics.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank.

In [ ]: