## HU Extension Assignment 08 E63 Big Data Analytics

### Handed out: 10/20/2017 Due by 4:00 PM EST on Saturday, 10/28/2017

If you are familiar with NLP API-s in languages other than Python or Python NLP API-s other than NLTK please be free to solve these problems using technology of your choice.

**Problem 1.** Use the text of the Universal Declaration of Human Rights (UDHR). Create a table for 5 languages in which you will collect statistics about the languages used. Place in that table the number of words in each language in UDHR, number of unique words, average length of words, number of sentences contained in UDHR and average number of words per sentence. Create a distribution of sentence lengths for each language. Plot those (non-cumulative) distributions on one diagram.

**(25%)**

**languages = ['English-Latin1', 'German\_Deutsch-Latin1','Hindi-UTF8','Bengali-UTF8','French\_Francais-Latin1']**

**for lang in languages :**

**num\_words = len(udhr.words(lang))**

**num\_uniq\_words = len(set(udhr.words(lang)))**

**num\_char\_words = len(udhr.raw(lang))**

**num\_sent = len(udhr.sents(lang))**

**print lang,num\_words,num\_uniq\_words,round(num\_char\_words/num\_words),num\_sent,round(num\_words/num\_sent)**

English-Latin1 1781 533 5.0 67 26.0

German\_Deutsch-Latin1 1521 579 6.0 60 25.0

Hindi-UTF8 2524 224 1.0 8 315.0

Bengali-UTF8 2591 235 1.0 3 863.0

French\_Francais-Latin1 1935 567 5.0 57 33.0

cfd=nltk.ConditionalFreqDist((lang,len(sent))

for lang in languages

for sent in udhr.sents(lang))

cfd.plot(cumulative=False)

****

Tried with another set of languages

from nltk.corpus import udhr

languages = ['Chickasaw', 'English', 'German\_Deutsch',

'Greenlandic\_Inuktikut', 'Hungarian\_Magyar', 'Ibibio\_Efik']

cfd = nltk.ConditionalFreqDist(

(lang, len(sent))

for lang in languages

for sent in udhr.sents(lang + '-Latin1'))

cfd.plot(cumulative=False)



**Problem 2**. Identify 10 most frequently used words longer than 7 characters in the entire corpus of Inaugural Addresses. Do not identify 10 words for every speech but rather 10 words for the entire corpus. Which among those words has the largest number of synonyms? List all synonyms for those 10 words. Which one of those 10 words has the largest number of hyponyms? List all hyponyms of those 10 most frequently used “long” words. The purpose of this problem is to familiarize you with WordNet and concepts of synonyms and hyponyms.

**(25%)**

Your literature for Problems 1 and 2 are chapters 1 and 2 of Natural Language Processing with Python book by Steven Bird et al.

**inaugural\_corpus=text4**

**word\_extract=[word.lower() for word in inaugural\_corpus if (len(word)>7 and word.isalpha())]**

**word\_frequency=FreqDist(word\_extract)**

**word\_frequency.tabulate(10)**

government citizens constitution national american congress interests political executive principles

593 237 205 154 147 129 113 106 97 93

**from nltk.corpus import wordnet as wn**

**for word in [x[0] for x in top\_10]:**

**for word\_syn in wordnet.synsets(word):**

**print word,word\_syn**

government Synset('government.n.01')

government Synset('government.n.02')

government Synset('government.n.03')

government Synset('politics.n.02')

citizens Synset('citizen.n.01')

constitution Synset('fundamental\_law.n.01')

constitution Synset('constitution.n.02')

constitution Synset('united\_states\_constitution.n.01')

constitution Synset('constitution.n.04')

constitution Synset('constitution.n.05')

national Synset('national.n.01')

national Synset('national.a.01')

national Synset('national.a.02')

national Synset('national.a.03')

national Synset('national.s.04')

national Synset('home.s.03')

national Synset('national.a.06')

national Synset('national.a.07')

american Synset('american.n.01')

american Synset('american\_english.n.01')

american Synset('american.n.03')

american Synset('american.a.01')

american Synset('american.a.02')

congress Synset('congress.n.01')

congress Synset('congress.n.02')

congress Synset('congress.n.03')

congress Synset('sexual\_intercourse.n.01')

interests Synset('interest.n.01')

interests Synset('sake.n.01')

interests Synset('interest.n.03')

interests Synset('interest.n.04')

interests Synset('interest.n.05')

interests Synset('interest.n.06')

interests Synset('pastime.n.01')

interests Synset('interest.v.01')

interests Synset('concern.v.02')

interests Synset('matter\_to.v.01')

political Synset('political.a.01')

political Synset('political.a.02')

political Synset('political.a.03')

executive Synset('executive.n.01')

executive Synset('executive.n.02')

executive Synset('administrator.n.03')

executive Synset('executive.a.01')

principles Synset('principle.n.01')

principles Synset('principle.n.02')

principles Synset('principle.n.03')

principles Synset('principle.n.04')

principles Synset('principle.n.05')

principles Synset('rationale.n.01')

#Plotting looks like interest has most synonyms

**cfd=nltk.ConditionalFreqDist((word,word\_syn)**

**for word in [x[0] for x in top\_10]**

**for word\_syn in wordnet.synsets(word))**

**cfd.plot(cumulative=True)**



#For hyponyms:

**for word in [x[0] for x in top\_10]:**

**for word\_syn in wordnet.synsets(word) :**

**print word,word\_syn.hyponyms**

government <bound method Synset.hyponyms of Synset('government.n.01')>

government <bound method Synset.hyponyms of Synset('government.n.02')>

government <bound method Synset.hyponyms of Synset('government.n.03')>

government <bound method Synset.hyponyms of Synset('politics.n.02')>

citizens <bound method Synset.hyponyms of Synset('citizen.n.01')>

constitution <bound method Synset.hyponyms of Synset('fundamental\_law.n.01')>

constitution <bound method Synset.hyponyms of Synset('constitution.n.02')>

constitution <bound method Synset.hyponyms of Synset('united\_states\_constitution.n.01')>

constitution <bound method Synset.hyponyms of Synset('constitution.n.04')>

constitution <bound method Synset.hyponyms of Synset('constitution.n.05')>

national <bound method Synset.hyponyms of Synset('national.n.01')>

national <bound method Synset.hyponyms of Synset('national.a.01')>

national <bound method Synset.hyponyms of Synset('national.a.02')>

national <bound method Synset.hyponyms of Synset('national.a.03')>

national <bound method Synset.hyponyms of Synset('national.s.04')>

national <bound method Synset.hyponyms of Synset('home.s.03')>

national <bound method Synset.hyponyms of Synset('national.a.06')>

national <bound method Synset.hyponyms of Synset('national.a.07')>

american <bound method Synset.hyponyms of Synset('american.n.01')>

american <bound method Synset.hyponyms of Synset('american\_english.n.01')>

american <bound method Synset.hyponyms of Synset('american.n.03')>

american <bound method Synset.hyponyms of Synset('american.a.01')>

american <bound method Synset.hyponyms of Synset('american.a.02')>

congress <bound method Synset.hyponyms of Synset('congress.n.01')>

congress <bound method Synset.hyponyms of Synset('congress.n.02')>

congress <bound method Synset.hyponyms of Synset('congress.n.03')>

congress <bound method Synset.hyponyms of Synset('sexual\_intercourse.n.01')>

interests <bound method Synset.hyponyms of Synset('interest.n.01')>

interests <bound method Synset.hyponyms of Synset('sake.n.01')>

interests <bound method Synset.hyponyms of Synset('interest.n.03')>

interests <bound method Synset.hyponyms of Synset('interest.n.04')>

interests <bound method Synset.hyponyms of Synset('interest.n.05')>

interests <bound method Synset.hyponyms of Synset('interest.n.06')>

interests <bound method Synset.hyponyms of Synset('pastime.n.01')>

interests <bound method Synset.hyponyms of Synset('interest.v.01')>

interests <bound method Synset.hyponyms of Synset('concern.v.02')>

interests <bound method Synset.hyponyms of Synset('matter\_to.v.01')>

political <bound method Synset.hyponyms of Synset('political.a.01')>

political <bound method Synset.hyponyms of Synset('political.a.02')>

political <bound method Synset.hyponyms of Synset('political.a.03')>

executive <bound method Synset.hyponyms of Synset('executive.n.01')>

executive <bound method Synset.hyponyms of Synset('executive.n.02')>

executive <bound method Synset.hyponyms of Synset('administrator.n.03')>

executive <bound method Synset.hyponyms of Synset('executive.a.01')>

principles <bound method Synset.hyponyms of Synset('principle.n.01')>

principles <bound method Synset.hyponyms of Synset('principle.n.02')>

principles <bound method Synset.hyponyms of Synset('principle.n.03')>

principles <bound method Synset.hyponyms of Synset('principle.n.04')>

principles <bound method Synset.hyponyms of Synset('principle.n.05')>

principles <bound method Synset.hyponyms of Synset('rationale.n.01')>

**#Interest also has highest number of hyponyms**

**cfd=nltk.ConditionalFreqDist((word,word\_syn.hyponyms)**

**for word in [x[0] for x in top\_10]**

**for word\_syn in wordnet.synsets(word))**

**cfd.plot(cumulative=True)**



**Problem 3.** Create your own grammar for the following sentence:

“Describe every step of your work and present all intermediate and final results in a Word document”.

**(10%)**

Your literature for Problem 3 is chapter 8 of Natural Language Processing with Python book by Steven Bird et al.

from nltk import CFG

sm\_grammar = CFG.fromstring("""

S -> NS Conj NS

NS -> V Adj N P DET N |V Adj N Conj N N P DET N N

V -> 'Describe'|'present'

N -> 'step'|'work'|'intermediate'|'final'|'results'|'Word'|'document'

Conj -> 'but'| 'and'

Adj -> 'every'|'step'| 'all'

P -> 'of'|'in'

DET -> 'your'|'a'

""")

rd\_parser = nltk.RecursiveDescentParser(sm\_grammar)

#text1 = nltk.word\_tokenize("Describe every step of your work and present all intermediate and final results in a Word document")

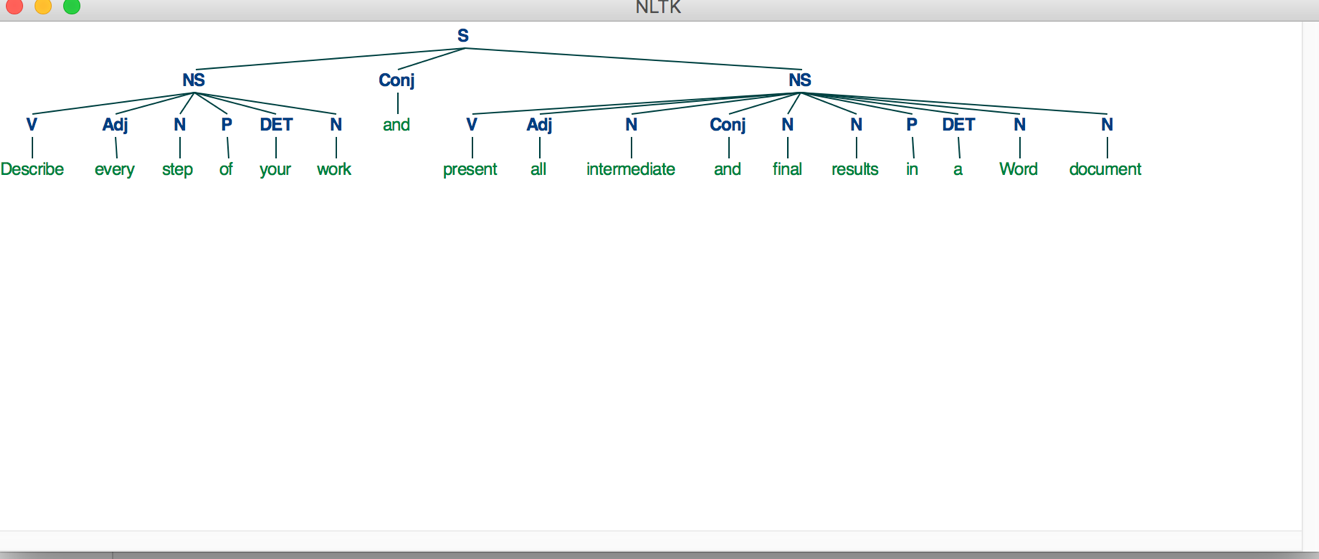
text1 = nltk.word\_tokenize("Describe every step of your work and present all intermediate and final results in a Word document")

#text = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']

for tree in rd\_parser.parse(text1):

print(tree)

tree.draw()



**Problem 4.** Install and compile Word2Vec C executables. Train CBOW model and create 200 dimensional embedding of Word Vectors. Demonstrate that you could run analogical reasoning when searching for country’s favorite food starting with japan and sushi. Note that words might have to be in lower case. Find favorite food for 5 different countries. Report improbable results as well as good results. Use scripts provided with original Google C code.

(**20%)**

**git clone https://github.com/William-Yeh/word2vec-mac.git**

Cloning into 'word2vec-mac'...

remote: Counting objects: 123, done.

remote: Total 123 (delta 0), reused 0 (delta 0), pack-reused 123

Receiving objects: 100% (123/123), 111.30 KiB | 0 bytes/s, done.

Resolving deltas: 100% (97/97), done.

bash-3.2$ **cd word2vec-mac/**

bash-3.2$ **ls -lrt**

total 1744

-rw-r--r--  1 smukherjee5  846622648   25294 Oct 28 02:24 word2vec.c

-rw-r--r--  1 smukherjee5  846622648    9386 Oct 28 02:24 word2phrase.c

-rw-r--r--  1 smukherjee5  846622648    4505 Oct 28 02:24 word-analogy.c

-rw-r--r--  1 smukherjee5  846622648  603955 Oct 28 02:24 questions-words.txt

-rw-r--r--  1 smukherjee5  846622648  168209 Oct 28 02:24 questions-phrases.txt

-rw-r--r--  1 smukherjee5  846622648     696 Oct 28 02:24 makefile

-rw-r--r--  1 smukherjee5  846622648    4398 Oct 28 02:24 distance.c

-rwxr-xr-x  1 smukherjee5  846622648     338 Oct 28 02:24 demo-word.sh

-rwxr-xr-x  1 smukherjee5  846622648     480 Oct 28 02:24 demo-word-accuracy.sh

-rwxr-xr-x  1 smukherjee5  846622648     437 Oct 28 02:24 demo-phrases.sh

-rwxr-xr-x  1 smukherjee5  846622648     911 Oct 28 02:24 demo-phrase-accuracy.sh

-rwxr-xr-x  1 smukherjee5  846622648     424 Oct 28 02:24 demo-classes.sh

-rwxr-xr-x  1 smukherjee5  846622648     703 Oct 28 02:24 demo-analogy.sh

-rw-r--r--  1 smukherjee5  846622648    5082 Oct 28 02:24 compute-accuracy.c

-rw-r--r--  1 smukherjee5  846622648    1209 Oct 28 02:24 README.original.txt

-rw-r--r--  1 smukherjee5  846622648    1089 Oct 28 02:24 README.md

-rw-r--r--  1 smukherjee5  846622648   11358 Oct 28 02:24 LICENSE

bash-3.2$ **make**

gcc word2vec.c -o word2vec -lm -pthread -Ofast -march=native -Wall -funroll-loops -Wno-unused-result

gcc word2phrase.c -o word2phrase -lm -pthread -Ofast -march=native -Wall -funroll-loops -Wno-unused-result

gcc distance.c -o distance -lm -pthread -Ofast -march=native -Wall -funroll-loops -Wno-unused-result

gcc word-analogy.c -o word-analogy -lm -pthread -Ofast -march=native -Wall -funroll-loops -Wno-unused-result

gcc compute-accuracy.c -o compute-accuracy -lm -pthread -Ofast -march=native -Wall -funroll-loops -Wno-unused-result

chmod +x \*.sh

#Downloaded the GoogleNews-vectors-negative300.bin dataset and trained it, and checked out favorite food for 5 countries. This was fun! Good results in green, not so good in red

**time ./word2vec -train GoogleNews-vectors-negative300.bin -output vectors.bin -cbow 0 -size 200 -window 5 -negative 0 -hs 1 -sample 1e-3 -threads 12 -binary 1**

Enter three words (EXIT to break): **japan sushi india**

Word: japan  Position in vocabulary: 165867

Word: sushi  Position in vocabulary: 23959

Word: india  Position in vocabulary: 68258

                                              Word              Distance

------------------------------------------------------------------------

                                         paranthas 0.543715

                                             idlis 0.536214

                                              dosa 0.524207

                                     gourmet\_pizza 0.515515

                                           sashimi 0.513758

                                     sushi\_sashimi 0.511360

                                           biryani 0.511333

                                       masala\_dosa 0.511041

                                   Bengali\_cuisine 0.509660

                                   tandoori\_dishes 0.507629

                                gourmet\_sandwiches 0.506948

                                      masala\_dosas 0.502800

                                             dosas 0.502265

                                           falooda 0.500842

                                         sushi\_bar 0.500413

                                          parantha 0.498633

                                            chaats 0.497123

                                Hyderabadi\_biryani 0.495566

                                             naans 0.495429

                                      nigiri\_sushi 0.494800

                                         idli\_dosa 0.494574

                                    Gujarati\_thali 0.493396

                                         pav\_bhaji 0.492637

                                        maki\_sushi 0.492372

                                     ghar\_ka\_khana 0.491226

                                   barbecued\_meats 0.490990

                                          biryanis 0.490579

                                      Thai\_curries 0.490037

                                           Mughlai 0.489993

                                          parathas 0.489660

                              mouthwatering\_dishes 0.489518

                                 thin\_crust\_pizzas 0.488992

                                             momos 0.488328

                                            paneer 0.487427

                                          pad\_thai 0.486535

                                           gourmet 0.485784

                                            nigiri 0.485342

                                         shawarmas 0.485050

                                              saag 0.485012

                                  chicken\_teriyaki 0.484684

**india paranthas america**

Word: india  Position in vocabulary: 68258

Word: paranthas  Position in vocabulary: 571227

Word: america  Position in vocabulary: 86028

                                              Word              Distance

------------------------------------------------------------------------

                                      shish\_kabobs 0.557459

                                 marinated\_veggies 0.551743

                                          parathas 0.545789

                            homemade\_peach\_cobbler 0.543386

                                   cranberry\_salad 0.533816

                                     ghar\_ka\_khana 0.533618

                                         sandwhich 0.532106

                                juicy\_cheeseburger 0.531192

                                      rajma\_chawal 0.530991

                                            kebabs 0.530630

                                   grilled\_skewers 0.530194

                                            kababs 0.525432

                                      veggie\_wraps 0.524753

                                        sandwiches 0.524587

                                      potato\_knish 0.524164

                                       lamb\_kabobs 0.523992

                                      crispy\_fries 0.522641

                                       sloppy\_Joes 0.521959

                                homemade\_guacamole 0.521521

                                   veggie\_sandwich 0.521482

                                      garlic\_toast 0.521361

                                            kebobs 0.521263

                                 strawberry\_crepes 0.520415

                                      french\_bread 0.519106

                                            chaats 0.518994

                                           bahn\_mi 0.518549

                              barbecue\_baked\_beans 0.518186

                                        baked\_tofu 0.517534

                                    banana\_muffins 0.517230

                                potatoes\_au\_gratin 0.516045

                                       sandwhiches 0.515675

                                          hog\_jowl 0.515658

                                    chicken\_kabobs 0.515647

                                   barbecued\_meats 0.515637

                                 homemade\_meatloaf 0.515431

                                        hush\_puppy 0.514153

                                             kabab 0.514031

                                Sweet\_potato\_fries 0.512398

                                    chickpea\_fries 0.512288

                          meatloaf\_mashed\_potatoes 0.512248

**america burger france**

Word: america  Position in vocabulary: 86028

Word: burger  Position in vocabulary: 19185

Word: france  Position in vocabulary: 225534

                                              Word              Distance

------------------------------------------------------------------------

                                           burgers 0.596398

                                      cheeseburger 0.567475

                                   Croque\_Monsieur 0.567335

                                     pommes\_frites 0.559768

                                            frites 0.559127

                                             fries 0.553552

                                             steak 0.551736

                                         hamburger 0.551258

                                         Cassoulet 0.544090

                                   croque\_monsieur 0.543765

                                     eggs\_benedict 0.543681

                                          sandwich 0.539229

                                            panino 0.537375

                                        L'\_Arpège 0.536452

                                     Eggs\_Benedict 0.535076

                                    gourmet\_burger 0.531478

                                      steak\_frites 0.526630

                                          baguette 0.524626

                                             tapas 0.521585

                                   seafood\_platter 0.520309

                                   quiche\_Lorraine 0.519704

                                        Coq\_au\_Vin 0.518634

                                           poutine 0.517821

                                      BLT\_sandwich 0.517422

                                            moules 0.516962

                                           Crêpes 0.515871

                                  sausage\_sandwich 0.514279

                                             sushi 0.512678

                                           burrito 0.511514

                                      crispy\_fries 0.511006

                                             boeuf 0.510991

                                bacon\_cheeseburger 0.510375

                                            jambon 0.509232

                                           Nicoise 0.508325

                                      plat\_du\_jour 0.508132

                                       bistro\_fare 0.507956

                                         croissant 0.507714

                                   Belgian\_waffles 0.507124

                                     sole\_meuniere 0.506795

                                  beef\_bourguignon 0.506109

**america burger srilanka**

Word: america  Position in vocabulary: 86028

Word: burger  Position in vocabulary: 19185

Word: srilanka  Position in vocabulary: 681172

                                              Word              Distance

------------------------------------------------------------------------

                                           burgers 0.605855

                                         hamburger 0.571108

                                      cheeseburger 0.548026

                                             fries 0.527229

                                              dosa 0.525214

                                             patty 0.522520

                                   seafood\_platter 0.520019

                                         pav\_bhaji 0.518482

                                    Butter\_Chicken 0.518420

                                     Chicken\_Tikka 0.517731

                                           paratha 0.512768

                                      crispy\_fries 0.511084

                                             sushi 0.510531

                                     veggie\_burger 0.510323

                                           burrito 0.508602

                                        naan\_bread 0.506946

                                        beef\_patty 0.506446

                                           Burgers 0.505923

                                      onion\_bhajis 0.505262

                                      Palak\_Paneer 0.504934

                                             dosas 0.503104

                                     chicken\_tikka 0.503024

                               roast\_beef\_sandwich 0.501978

                                   Reuben\_sandwich 0.501729

                                          Jalfrezi 0.501301

                                       seekh\_kebab 0.500758

                                    gourmet\_burger 0.500300

                                Hyderabadi\_biryani 0.499173

                                            kababs 0.499042

                                          pub\_grub 0.496828

                                  Tandoori\_Chicken 0.495762

                                            samosa 0.495572

                                          sandwich 0.495037

                                          vindaloo 0.494824

                                         poppadoms 0.494633

                                bacon\_cheeseburger 0.494166

                                          Biriyani 0.493329

                                          Tandoori 0.492739

                                  chilli\_con\_carne 0.492610

                                              Idli 0.492012

**america burger thai**

Word: america  Position in vocabulary: 86028

Word: burger  Position in vocabulary: 19185

Word: thai  Position in vocabulary: 310551

                                              Word              Distance

------------------------------------------------------------------------

                                             sushi 0.605126

                                           burgers 0.600997

                                          Pad\_Thai 0.584346

                                           Banh\_Mi 0.564475

                                      noodle\_salad 0.560179

                                      cheeseburger 0.558221

                                  Kung\_Pao\_chicken 0.557719

                                      Spicy\_Shrimp 0.556801

                                             Satay 0.556700

                                           burrito 0.555153

                                     eggs\_benedict 0.554262

                                     crab\_rangoons 0.551059

                                             Ponzu 0.550786

                                      papaya\_salad 0.548439

                                             Sushi 0.545634

                                        Noodle\_Bar 0.545430

                                   Reuben\_sandwich 0.543165

                                  Tandoori\_Chicken 0.542567

                                     gourmet\_pizza 0.542190

                                 steamed\_dumplings 0.542173

                                     Chicken\_Satay 0.538203

                            Hainanese\_chicken\_rice 0.537499

                                          pad\_thai 0.535955

                                           som\_tam 0.535822

                                     tom\_yum\_goong 0.535579

                                           tom\_yum 0.535117

                                     satay\_chicken 0.535020

                                        mee\_goreng 0.534964

                                         shawarmas 0.534697

                                    sesame\_chicken 0.534365

                                       Shabu\_Shabu 0.534132

                                       noodle\_dish 0.533688

                                        Beef\_Salad 0.533450

                                       Lemon\_Grass 0.532950

                                  beef\_noodle\_soup 0.532649

                                    appetizer\_menu 0.532427

                                   chicken\_lo\_mein 0.532378

                                             Spicy 0.532348

                                 scallion\_pancakes 0.531948

                                            dimsum 0.531204

**Problem 5.** Install and run Genism Python Word2Vec API. Find the most probable words you will obtain when you start with an emperor add a woman and subtract a man. Use this tutorial as a guide <https://rare-technologies.com/word2vec-tutorial/>

**(20%)**

**conda install gensim**

Fetching package metadata ...........

Solving package specifications: .

Package plan for installation in environment /Users/smukherjee5/anaconda2:

The following NEW packages will be INSTALLED:

    bz2file:    0.98-py27\_0

    gensim:     3.0.1-py27hb8c1596\_0

    smart\_open: 1.5.3-py27\_0

Proceed ([y]/n)? y

bz2file-0.98-p 100% |##############################################################################################################################################################################################| Time: 0:00:00 271.69 kB/s

smart\_open-1.5 100% |##############################################################################################################################################################################################| Time: 0:00:00 630.76 kB/s

gensim-3.0.1-p 100% |##############################################################################################################################################################################################| Time: 0:00:02   4.35 MB/s

**import gensim, logging**

**logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)**

**model = gensim.models.KeyedVectors.load\_word2vec\_format('GoogleNews-vectors-negative300.bin',binary=True)**

2017-10-28 08:52:26,045 : INFO : loading projection weights from GoogleNews-vectors-negative300.bin

2017-10-28 08:53:35,665 : INFO : loaded (3000000, 300) matrix from GoogleNews-vectors-negative300.bin

**most\_similar = model.most\_similar(positive=['emperor', 'woman'],negative=['man'])**

**print(most\_similar)**

[(u'empress', 0.6470329165458679), (u'emperors', 0.6074110269546509), (u'Emperor', 0.5940502882003784), (u'empresses', 0.5697327852249146), (u'Wu\_Zetian', 0.5530362129211426), (u'imperial\_throne', 0.5474318265914917), (u'Empress\_Dowager\_Cixi', 0.5443990230560303), (u'Queen\_Consort', 0.5418635606765747), (u'imperial', 0.5319660902023315), (u'Imperial\_Household', 0.5288584232330322)]

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document with explanations of the purpose of those commands. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. For issues and comments visit the class Discussion Board. If you use some other language other than Python in your daily work with NLP, please be free to use that language and a framework of your choice to do this assignment.