homework for nlp 1

members:

1. `Li Ruiyuan`
2. `Zhang Xuhao`

excise 1: Mophology

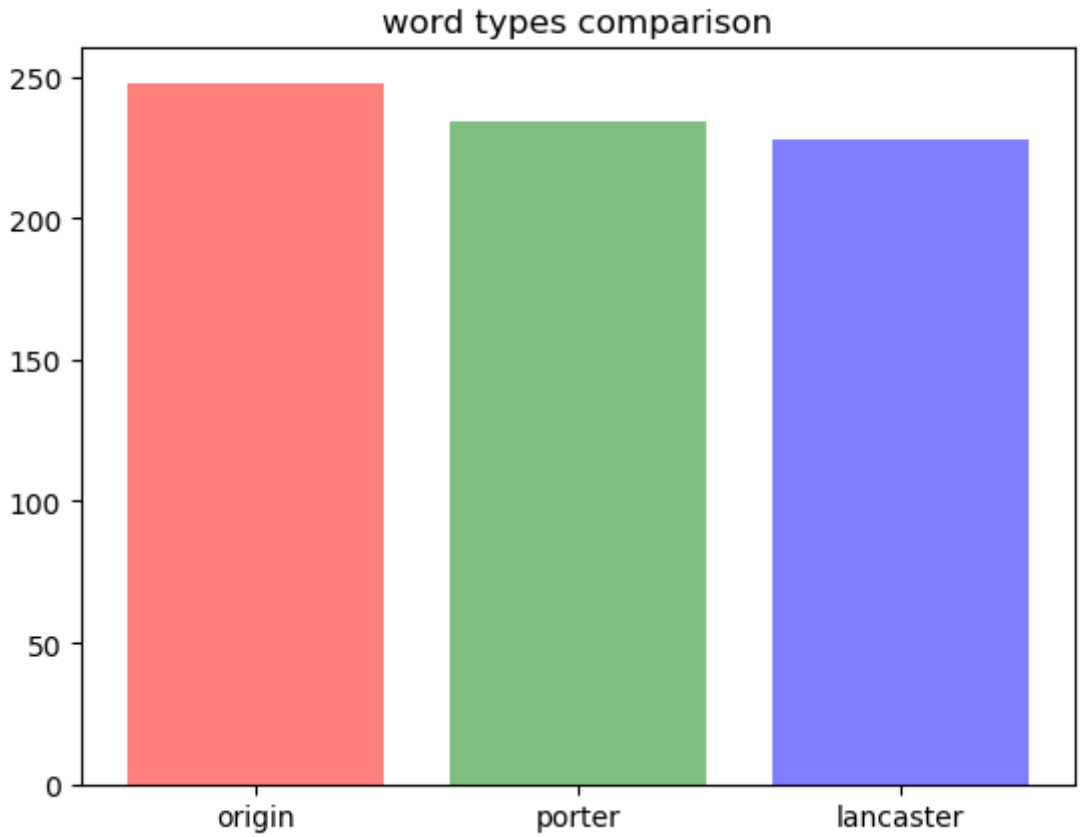
- inflection
  - could->can
  - Imitating->imitate
  - Uses->use
- Derivation
  - actually->actual
  - Successful->success
  - Operation->operate
- Compounding
  - natural language processing
  - Text patterns
  - Text normalization
- Cliticisation
  - I’m
  - We’ll
  - User’s

exercise 2: stemming

2a

N=248
after stemming:

```
porter=234
lancaster=228
```



2b

omission

psychotherapist
would
user's

comission

- weizenbaum -> weizenba (porter,lancaster)
  this is actually a name
- used->us (lancaster)
- care->car(lancaster)
- despite->despit(porter,lancaster)
- sometimes->sometim(porter,lancaster)
  the stemmers just remove suffixes mechanically, including all the -ate, -e, -es, -s, -nize...
  PorterStemmer started in the 1980s and its main concern is to delete the common endings of words so that they can be parsed into generic forms. It's not too complicated, and its development stopped.

While Lancaster's algorithm is more radical, sometimes the results will be strange. And both of them are not performing well in terms of cliticisation.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
import matplotlib.pyplot as plt
import matplotlib as mpl
wordnet_lemmatizer = WordNetLemmatizer()
#nltk.download()
porter=PorterStemmer()
lancaster=LancasterStemmer()
raw_list=[]
with open('sorted_types.txt','r') as f:
    line = f.read().strip('\n')
    raw_list = line.split("\n")

print(raw_list)
# output=[]
# for i in raw:
#     output.append(PorterStemmer.stem(i))
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster Stemmer"))
for word in raw_list:
    print("{0:20}{1:20}{2:20}".format(word,porter.stem(word),lancaster.stem(word)))
#nltk.corpus.gutenberg.fileids()

porter_list = []
lancaster_list = []
dic_porter = {}
dic_lancaster = {}
for word in raw_list:
    porter_list.append(porter.stem(word))
    lancaster_list.append(lancaster.stem(word))
    dic_porter[porter.stem(word)] =porter.stem(word)
    dic_lancaster[lancaster.stem(word)] =lancaster.stem(word)

name_list=["origin","porter","lancaster"]
num_list=[len(raw_list),len(dic_porter),len(dic_lancaster)]
print(num_list)
plt.bar(range(len(num_list)), num_list, color='rgb', tick_label=name_list,alpha=0.5)
plt.title('word types comparison')
plt.show()
```

exercise 3: lemmatization

version1

using `from nltk.stem import WordNetLemmatizer`

3a

lemmalized=242.

3b

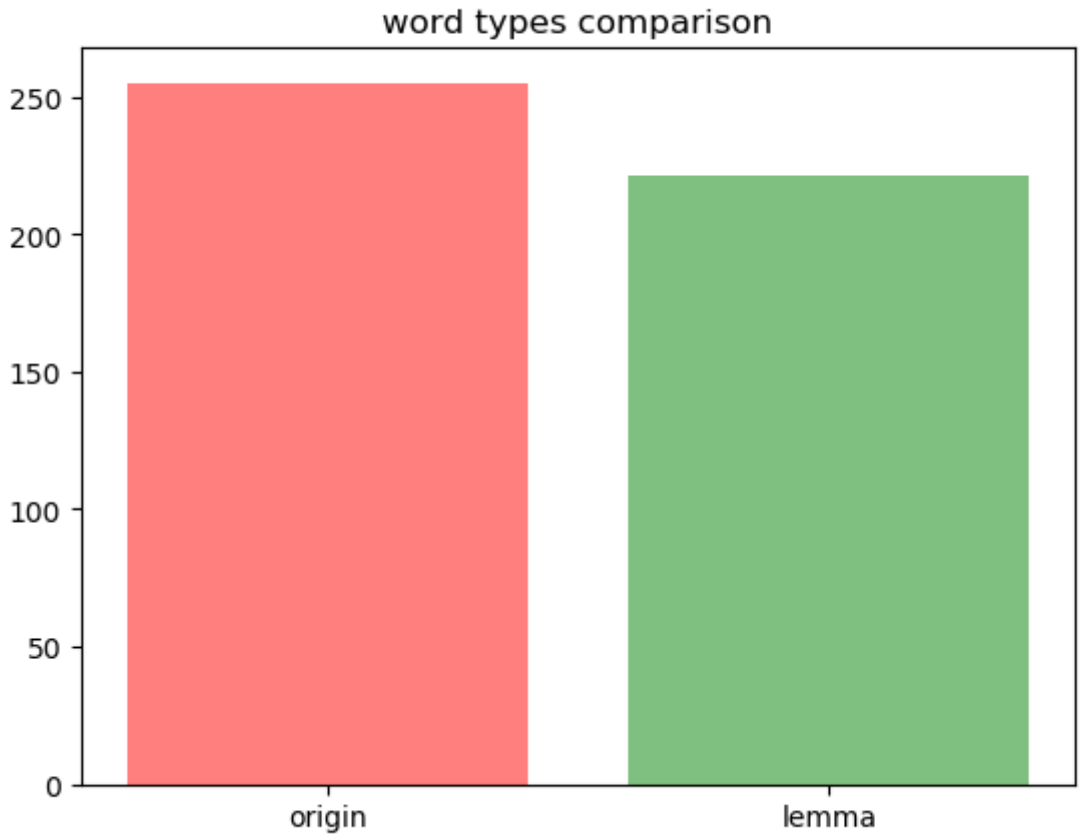
- an an
  'an' should be lemmatized as 'a',it is an omission
- as a
  'as'shouldn't be lemmatized, it is a comission
- converting converting
  'converting'shouldn be lemmatized as convert, it is an omission

version2

using `spacy and model:en_core_web_sm`

3a

origin=255, lemmalized=221



3b

- would would
  'would' should be lemmatized as 'will',it is an omission
- separating separating
  'separating' should be lemmatized as 'separate',it is an omission
- relies relie
  'relies' should be lemmatized as 'rely',it is a rule problem.

exercise 4

4a

- Character duplication：
  - Hummm
  - ZZZZZZZZZZZzzzzzzzzzzzzzzzz
  - Errrrrrrrr
- Number transliteration：
  - man2

    - b4
    - humm7
- Character deletion：
    - Juz
    - Wat
    - Abt
- Character replacement：
    - Feekz
    - Alvin
    - faiz
- Character transposition:
    - Fisrt
    - Soprts
    - Tasek
- Phonetic substitution:
    - bla

4b

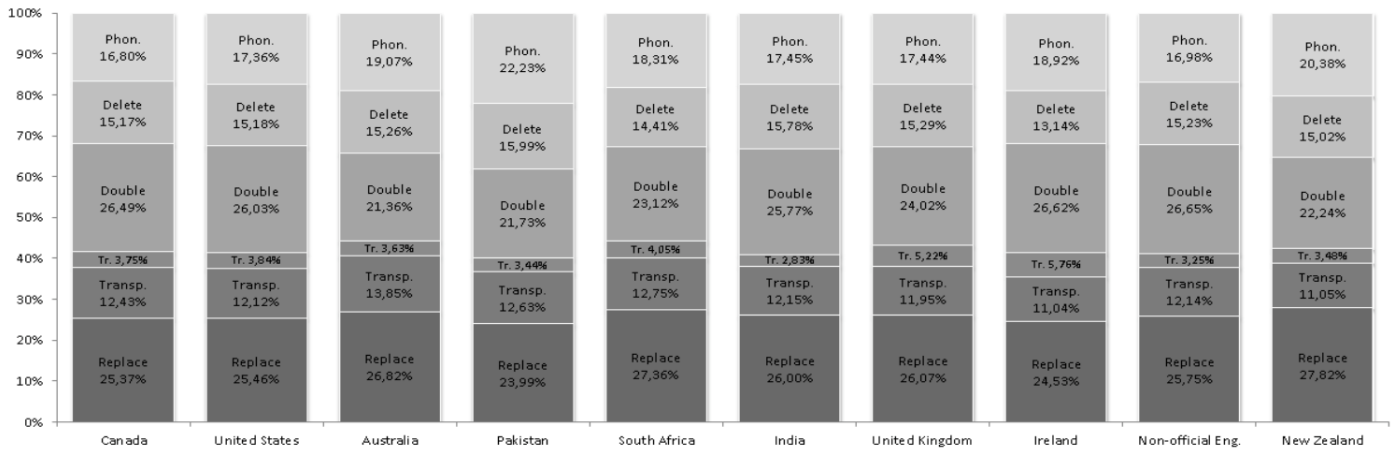


Figure 2: Transformation trends by English-speaking countries.

> Character duplication about 26%
> Number transliteration about 4%
> Character deletion about 15%
> Character replacement about 25%
> Character transposition about 12.5%
> Phonetic substitution about 17%

4c

Text also reflects the demographic characteristics of the writer (or speaker): their age, gender, race, socioeconomic class can all influence the linguistic properties of the text we are processing.

> **This blog in M-train109.txt:**
> *“I’m sorry if Ryuzuki seemed too worthless when you talked to him a while ago, I’ll make up for that. He sometimes forgets his manners around someone he supposedly knows very well. Did I mention just how special this kid is? He’s been favored by my two sisters, the angels of life and live, and he’s blessed by the Egyptian god Ra . If you don’t believe me, ask him to show you his right hand when you talk to him. If you can still read katakana, you’ll find something very interesting on his hand. He’s constantly in good health, and he never gets hurt by love. Besides, he already found Sakura. According to him, he doesn’t care if he never gets someone he can call his own, as long as Sakura is there. He cares a lot for her, and I don’t want to hurt him or Sakura because of that. Ask him yourself, if you wish. He actually does things he’d never to with other girls with her, like sing, try to dance, and show her just how much he really tries at stuff. Don’t worry about her getting hurt by him or me, since I can’t hurt her or you, and you know that. I’m really sorry if you thought I would hurt her, even in my full glory I will not try to. I am still of God, and as long as you both are, you can expect me to protect you instead of hurting you. (Shi wa Tenshi)”*

It looks like the blog was written by a woman, and has a religious color. It can be seen that he should believe in a certain sect. Indeed, these blogs will reflect many habits and characteristics of the author.

exercise 5

5a

Use python and Regular expressions.
Answer:

```
M:
[('1s', 65), ('2n', 52), ('4t', 48), ('0t', 48), ('3r', 41), ('5t', 38), ('0m', 38), ('6t', 32), ('0p', 32), ('0a',
32), ('p3', 29), ('8t', 28), ('0s', 28), ('2m', 21), ('9t', 20), ('2d', 16), ('1a', 14), ('b4', 14), ('7t', 14), ('
5p', 12), ('2g', 12), ('0k', 12), ('9a', 11), ('1p', 11), ('5a', 11), ('2p', 11), ('1t', 10), ('5m', 10), ('g5', 9)
, ('5k', 8), ('3t', 8), ('2a', 8), ('e4', 8), ('h0', 7), ('7p', 7), ('2t', 7), ('g3', 7), ('3p', 6), ('7a', 6), ('6
p', 6), ('s2', 6), ('8r', 6), ('9p', 6), ('8a', 5), ('3s', 5), ('8p', 5), ('y1', 5), ('2s', 5), ('0i', 4), ('i1', 4
), ('0r', 4), ('4p', 4), ('s1', 4), ('y2', 4), ('0h', 4), ('x1', 4), ('3a', 4), ('4g', 4), ('o1', 4), ('n9', 4), ('
p1', 4), ('g1', 4), ('g2', 4), ('g4', 4), ('2k', 3), ('n2', 3), ('0l', 3), ('4k', 3), ('x4', 3), ('0n', 3), ('l1',
3), ('k8', 3), ('x0', 3), ('9k', 3), ('6a', 3), ('4n', 3), ('r0', 3), ('m1', 3), ('r1', 3), ('x3', 3), ('8e', 3), (
'5i', 3), ('p2', 3), ('1n', 3), ('5c', 2), ('g8', 2), ('4w', 2), ('0b', 2), ('1k', 2), ('e0', 2), ('y7', 2), ('0c',
2), ('x2', 2), ('d5', 2), ('2e', 2), ('8s', 2), ('6s', 2), ('5e', 2), ('4h', 2), ('k5', 2), ('8m', 2), ('1m', 2), (
'm7', 2), ('6k', 2), ('e7', 2), ('r4', 2), ('4l', 2), ('n6', 2), ('4f', 2), ('0g', 2), ('t2', 2), ('4r', 2), ('g6',
2), ('2c', 1), ('6f', 1), ('v2', 1), ('3m', 1), ('0f', 1), ('7x', 1), ('9g', 1), ('i3', 1), ('r8', 1), ('w3', 1), (
'v7', 1), ('r2', 1), ('7k', 1), ('p4', 1), ('w8', 1), ('w1', 1), ('6e', 1), ('e2', 1), ('4x', 1), ('4i', 1), ('m3',
1), ('4s', 1), ('j2', 1), ('h1', 1), ('v9', 1), ('3g', 1), ('m6', 1), ('1g', 1), ('x8', 1), ('d3', 1), ('x7', 1), (
'j0', 1), ('9i', 1), ('h2', 1), ('2b', 1), ('3h', 1), ('k0', 1), ('2o', 1), ('0e', 1), ('1u', 1), ('c1', 1), ('3d',
1), ('0w', 1), ('t1', 1), ('8c', 1), ('i2', 1), ('8k', 1), ('0u', 1), ('5o', 1), ('5h', 1), ('9e', 1), ('4c', 1), (
'm2', 1), ('z0', 1), ('7s', 1), ('f3', 1), ('1e', 1), ('k2', 1), ('4a', 1), ('e1', 1), ('2h', 1), ('b2', 1), ('s0',
1), ('4y', 1), ('2w', 1), ('5s', 1), ('4e', 1), ('t0', 1)]
F:
[('1s', 107), ('2d', 55), ('4t', 54), ('0p', 52), ('2n', 47), ('6t', 43), ('3r', 42), ('b4', 41), ('0a', 38), ('0t'
, 36), ('8t', 35), ('7t', 29), ('y1', 25), ('5t', 25), ('k1', 24), ('9t', 23), ('2m', 23), ('2g', 20), ('t0', 18),
('0s', 17), ('2x', 15), ('m1', 14), ('0m', 13), ('1p', 12), ('5p', 12), ('3t', 11), ('3a', 10), ('4p', 10), ('o1',
9), ('n2', 9), ('9p', 9), ('7a', 9), ('8e', 9), ('e1', 9), ('5m', 9), ('d2', 9), ('5a', 8), ('2a', 8), ('6a', 8), (
'2p', 8), ('8r', 8), ('8a', 7), ('1a', 7), ('2t', 7), ('4e', 7), ('2l', 7), ('g7', 7), ('0n', 6), ('1w', 6), ('1t',
6), ('8p', 6), ('3p', 6), ('0k', 6), ('6p', 6), ('0i', 6), ('4w', 5), ('i2', 5), ('4r', 5), ('n6', 4), ('5i', 4), (
'h0', 4), ('4g', 4), ('e3', 4), ('m3', 4), ('9m', 4), ('2h', 4), ('4n', 4), ('0u', 4), ('n7', 4), ('a2', 4), ('n9',
3), ('8i', 3), ('0b', 3), ('2b', 3), ('5c', 3), ('3s', 3), ('6i', 3), ('2y', 3), ('0y', 3), ('b8', 3), ('2k', 3), (
'h8', 3), ('4y', 3), ('r3', 3), ('0f', 2), ('b9', 2), ('3x', 2), ('2i', 2), ('5x', 2), ('r8', 2), ('a1', 2), ('7o',
2), ('3m', 2), ('f8', 2), ('3d', 2), ('r2', 2), ('j0', 2), ('k2', 2), ('h2', 2), ('y2', 2), ('m2', 2), ('m8', 2), (
'x5', 2), ('e2', 2), ('k5', 2), ('0c', 2), ('h1', 2), ('t2', 2), ('7g', 2), ('s2', 2), ('3n', 2), ('h7', 2), ('h3',
2), ('0w', 2), ('4l', 2), ('u5', 2), ('k0', 2), ('2o', 2), ('w2', 2), ('p5', 2), ('3e', 2), ('6f', 2), ('5q', 2), (
'1h', 2), ('9a', 2), ('9s', 2), ('c2', 1), ('1k', 1), ('4x', 1), ('2s', 1), ('1i', 1), ('7i', 1), ('s0', 1), ('0o',
1), ('a8', 1), ('3i', 1), ('4m', 1), ('p1', 1), ('b5', 1), ('r1', 1), ('7f', 1), ('9f', 1), ('c6', 1), ('9i', 1), (
'g1', 1), ('8y', 1), ('2c', 1), ('x6', 1), ('x4', 1), ('2w', 1), ('4o', 1), ('6w', 1), ('5b', 1), ('t1', 1), ('k3',
1), ('c4', 1), ('3y', 1), ('r0', 1), ('f1', 1), ('8m', 1), ('6j', 1), ('4a', 1), ('7p', 1), ('n3', 1), ('s3', 1), (
'u1', 1), ('9c', 1), ('2f', 1), ('g3', 1), ('3l', 1), ('l3', 1), ('y5', 1), ('h4', 1), ('7h', 1), ('u7', 1), ('c3',
1), ('v5', 1), ('0r', 1), ('4k', 1), ('g2', 1), ('a7', 1), ('6l', 1), ('0l', 1), ('1m', 1), ('0h', 1), ('c0', 1), (
'y6', 1), ('d1', 1), ('4f', 1), ('4h', 1), ('5l', 1), ('6o', 1), ('8k', 1), ('3q', 1), ('n1', 1), ('w0', 1), ('3b',
1), ('2e', 1), ('3k', 1), ('4b', 1), ('1c', 1), ('r9', 1), ('1l', 1), ('7s', 1), ('5k', 1), ('a6', 1), ('8o', 1), (
'8s', 1), ('l2', 1)]
```

5b

As shown in the figure above, the 5 longest words used by male are **1s 2n 4t 0t 3r**
female
1s 2d 4t 0p 2n

1s maybe means ones

5c

We found that the numbers 1, 2 and 4 are often used for transliteration。
like：

- b4->before
- m2->me too

maybe female is more likely to use 2 and male prefer 3. As for number 1， both male and female like to use it.

5d

method:

1. Build a replacement table sub['eight','ate','at','ight'...],as there are usages like
   `l8er。 what have i done? tlk l8er. m.c`
2. get all the txt material by `f.read().strip('\n').split(" ")`
3. store words in a list `raw_list`
4. judge if the i*th* token is a figure or a car/ID/street number
5. use every words in `sub[]` to replace 8
6. compare the new word with wods in dictionary, to assure it is a real word
7. according to the right rate, change the position of words in `sub[]`
8. `PyEnchant` can give some new words to a wrong answer
   using `d.suggest(word)`

experience:

First of all, when we encountered this problem, we were not full of solutions and feasible plans, because the two members in our group are both from China, and English was not our native language. We were especially unfamiliar with internet phrases.

The language using in blog writing usually contains a lot of slang of local culture or short-term popular witticism, which has strong regional features and timeliness, so it was quite difficult for us.

Therefore, after we find out the phrase of number substitution, the biggest problem is to restore its original meaning. For the most part, we relied on guesswork or seeked help from Google. This took many hours.

We chose python here because it is more convenient than other applications, and there is no need to download and install it online. All we need is import re, and the flexibility of python is much better than other programs. We can read the file in the way we want, without being limited to the mode set by others.