Homework 1

Sophie Du

In [3]:
```
import nltk
nltk.download()
import spacy
from spacy.lang.en import English
import stanfordnlp
stanfordnlp.download('en')
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/
index.xml
Using the default treebank "en_ewt" for language "en".
Would you like to download the models for: en_ewt now? (Y/n)
Y

Default download directory: /Users/Sophie/stanfordnlp_resources
Hit enter to continue or type an alternate directory.


Downloading models for: en_ewt
Download location: /Users/Sophie/stanfordnlp_resources/en_ewt_models.zi
p

100%|██████████| 235M/235M [01:33<00:00, 2.51MB/s]

Download complete.  Models saved to: /Users/Sophie/stanfordnlp_resource
s/en_ewt_models.zip
Extracting models file for: en_ewt
Cleaning up...Done.
```

Problem 1

In [4]:
```
import os
import time
with open('20-newsgroups/sci.med.txt', encoding = 'utf8', errors = 'igno
re') as f:
    text = str(f.read().replace('\n', ''))
```

```
In [5]: ## sample text data
        sample = """Newsgroup: sci.med
        document_id: 58045
        From: fulk@cs.rochester.edu (Mark Fulk)
        Subject: Re: Breech Baby Info Needed

        In article <1993Apr5.151818.27409@trentu.ca> xtkmg@trentu.ca (Kate Grego
        ry) writes:
        >In article <1993Apr3.161757.19612@cs.rochester.edu> fulk@cs.rochester.e
        du (Mark Fulk) writes:
        >>
        >>Another uncommon problem is maternal hemorrhage.  I don't remember the
        >>incidence, but it is something like 1 in 1,000 or 10,000 births.  It i
        s hard
        >>to see how you could handle it at home, and you wouldn't have very muc
        h time.
        >>
        >>thing you might consider is that people's risk tradeoffs vary.  I cons
        ider
        >>a 1/1,000 risk of loss of a loved one to require considerable effort i
        n
        >>the avoiding.
        >
        >Mark, you seem to be terrified of the birth process

        That's ridiculous!

        >and unable to
        >believe that women's bodies are actually designed to do it.

        They aren't designed, they evolved.  And, much as it discomforts us, in
        humans a trouble-free birth process was sacrificed to increased brain an
        d
        cranial size.  Wild animals have a much easier time with birth than huma
        ns do.
        Domestic horses and cows typically have a worse time.  To give you an id
        ea:
        my family tree is complicated because a few of my pioneer great-great-
        grandfathers had several wives, and we never could figure out which wife
        had each child.  One might ask why this happened.  My great-great-
        grandfathers were, by the time they reached their forties, quite prosper
        ous
        farmers.  Nonetheless, they lost several wives each to the rigors of
        childbirth; the graveyards in Spencer, Indiana, and Boswell, North Dakot
        a,
        contain quite a few gravestones like "Ida, wf. of Jacob Liptrap, and
        baby, May 6, 1853."

        >You wanted
        >to section all women carrying breech in case one in a hundred or a
        >thousand breech babies get hung up in second stage,

        More like one in ten.  And the consequences can be devastating; I have
        direct experience of more than a dozen victims of a fouled-up breech bir
        th.
```

>and now you want
>all babies born in hospital based on a guess of how likely maternal
>hemorrhage is and a false belief that it is fatal.

It isn't always fatal. But it is often fatal, when it happens out of
reach of adequate help. More often, it permanently damages one's healt
h.

Clearly women's bodies _evolved_ to give birth (I am no believer in divi
ne
design); however, evolution did not favor trouble-free births for human
s.

>You have your kids where you want. You encourage your wife to
>get six inch holes cut through her stomach muscles, expose herself
>to anesthesia and infection, and whatever other "just in case" measures
>you think are necessary.

My, aren't we wroth! I haven't read a more outrageous straw man attack
in months! I can practically see your mouth foam.

We're statistically sophisticated enough to balance the risks. Although
I can't produce exact statistics 5 years after the last time we looked
them up, rest assured that we balanced C-section risks against other ris
ks.
I wouldn't encourage my wife to have a Caesarean unless it was clearly
indicated; on the other hand, I am opposed (on obvious grounds) to waiti
ng
until an emergency to give in.

And bear this in mind: my wife took the lead in all of these decisions.
We talked things over, and I did a lot of the leg work, but the main
decisions were really hers.

>But I for one am bothered by your continued
>suggestions, especially to the misc.kidders pregnant for the first
>time, that birth is dangerous, even fatal, and that all these
>unpleasant things are far better than the risks you run just doing
>it naturally.

I don't know of very many home birth advocates, even, that think that
a first-time mother should have her baby at home.

>I'm no Luddite. I've had a section. I'm planning a hospital birth
>this time. But for heaven's sake, not everyone needs that!

But people should bother to find out the relative risks. My wife was
unwilling to take any significant risks in order to have nice surroundin
gs.
In view of the intensity of the birth experience, I doubt surroundings
have much importance anyway. Somehow the values you're advocating seem
all lopsided to me: taking risks, even if fairly small, of serious
permanent harm in order to preserve something that is, after all,
an esthetic consideration.
--
Mark A. Fulk   University of Rochester

```
Computer Science Department     fulk@cs.rochester.edu
"""
```

NLTK

In [6]:
```
## Sentence tokenize
from nltk.tokenize import sent_tokenize
now = time.time()
sent_nltk = sent_tokenize(text)
print('NLTK took %s seconds for sentence tokenize'%(time.time()-now))
```

```
NLTK took 1.4176156520843506 seconds for sentence tokenize
```

In [8]:
```
## Word tokenize
from nltk.tokenize import word_tokenize
## tokenize text
now = time.time()
word_nltk = word_tokenize(text)
print('NLTK took %s seconds for word tokenize'%(time.time()-now))
```

```
NLTK took 4.894775867462158 seconds for word tokenize
```

In [15]:
```
## stemming
from nltk.stem import PorterStemmer
ps = PorterStemmer()
words = list(set(word_tokenize(sample)))
stemmed = []
for word in words:
    stemmed.append(ps.stem(word))
stemmed[:10]
```

Out[15]:
```
['everyon',
 'through',
 'import',
 'mind',
 'direct',
 'happen',
 'howev',
 'damag',
 'but',
 'like']
```

In [16]:
```
## POS tagging
nltk.pos_tag(words)[:10]
```

Out[16]:
```
[('everyone', 'NN'),
 ('through', 'IN'),
 ('importance', 'NN'),
 ('mind', 'NN'),
 ('direct', 'JJ'),
 ('happens', 'VBZ'),
 ('however', 'RB'),
 ('damages', 'NNS'),
 ('but', 'CC'),
 ('likely', 'JJ')]
```

Spacy

```
In [88]: ## sentence tokenize
         now = time.time()
         nlp = English()
         nlp.max_length = 20000000
         nlp.add_pipe(nlp.create_pipe('sentencizer'))

         now = time.time()
         doc = nlp(text)
         sent_spc = [sent.string.strip() for sent in doc.sents]
         print('Spacy took %s seconds for sentence tokenize'%(time.time()-now))

         Spacy took 8.43147897720337 seconds for sentence tokenize
```

```
In [90]: ## word tokenize
         nlp = English()
         nlp.max_length = 50000000
         now = time.time()
         doc = nlp(text)
         word_spc = [token.text for token in doc]
         print('Spacy took %s seconds for word tokenize'%(time.time()-now))

         Spacy took 7.93451714515686 seconds for word tokenize
```

```
In [17]: ## stemming
         sp = spacy.load('en')
         sp.max_length = 50000000
         doc_sp = sp(text)
         lemmas = [token.lemma_ for token in doc_sp]
         lemmas[:10]

Out[17]: ['newsgroup',
          ':',
          'sci.meddocument_id',
          ':',
          '57110from',
          ':',
          'bed@intacc.uucp',
          '(',
          'Deb',
          'waddington)subject']
```

```
In [19]: ## POS tagging
         pos_tag = []
         for token in doc_sp:
             pos_tag.append([token, token.pos_])
         pos_tag[:10]
```

```
Out[19]: [[Newsgroup, 'NOUN'],
          [:, 'PUNCT'],
          [sci.meddocument_id, 'X'],
          [:, 'PUNCT'],
          [57110From, 'NUM'],
          [:, 'PUNCT'],
          [bed@intacc.uucp, 'PROPN'],
          [(, 'PUNCT'],
          [Deb, 'PROPN'],
          [Waddington)Subject, 'NOUN']]
```

Parallelization of tokenization

```
In [21]: from joblib import Parallel, delayed
```

```
In [22]: sentences = sent_tokenize(text); sentences[:5]
```

```
Out[22]: ["Newsgroup: sci.meddocument_id: 57110From: bed@intacc.uucp (Deb Waddin
         gton)Subject: INFO NEEDED: Gaucher's DiseaseI have a 42 yr old male fri
         end, misdiagnosed as having osteopporosis for two years, who recently f
         ound out that his illness is the rare Gaucher's disease.",
          "Gaucher's disease symptoms include: brittle bones (he lost 9  inches
         off his hieght); enlarged liver and spleen; internal bleeding; and fati
         gue (all the time).",
          'The problem (in Type 1) is attributed to a genetic mutation where the
         re is a lack of the enzyme glucocerebroside in macrophages so the cells
         swell up.',
          'This will eventually cause death.Enyzme replacement therapy has been
         successfully developed and approved by the FDA in the last few years so
         that those patients administered with this drug (called Ceredase) repor
         t a remarkable improvement in their condition.',
          'Ceredase, which is manufactured by biotech biggy company--Genzyme--co
         sts the patient $380,000 per year.']
```

```
In [113]: ## nltk word tokenize
          now = time.time()
          x1 = Parallel(n_jobs = 3)(delayed(word_tokenize)(sent) for sent in sente
          nces)
          print('NLTK took %s seconds for word tokenize with joblib'%(time.time()-
          now))
```

```
          NLTK took 7.841717958450317 seconds for word tokenize with joblib
```

In [123]:
```
## spacy word tokenize
now = time.time()
for doc in sp.pipe(sentences, batch_size = 5000, n_threads = 3):
    [token.text for token in doc]

print('Spacy took %s seconds for word tokenize with n_thread'%(time.time
() - now))
```

```
Spacy took 108.841796875 seconds for word tokenize with n_thread
```

Problem 2

In [24]:
```
import re
```

In [25]:
```
## match all emails in text and compile a set of all found email address
emails = []
emails_reg = re.findall('[a-zA-Z0-9_.]+@[a-zA-Z0-9.]+', text)
emails.extend(emails_reg)
emails = list(set(emails))
```

In [26]:
```
## find all dates in text (e.g. 04/12/2019, April 20th 2019, etc)
dates = []
## mm/dd/yy
reg1 = re.findall(r'[0-9]{2}/[0-9]{2}/[0-9]{2}', text)
## dd Month yyyy
reg2 = re.findall(r"[\d]{1,2}[th]? [ADFJMNOS]\w* [\d]{4}", text)
## Month dd yyyy
reg3 = re.findall(r"[ADFJMNOS]\w* [\d]{1,2}[\,]? [\d]{4}", text)
## yyyy/mm/dd
reg4 = re.findall(r"[\d]{4}/\w*/[\d]{1,2}", text)
## yyyyMonthdd
reg5 = re.findall(r"[\d]{4}[ADFJMNOS]\w*[\d]{1,2}", text)

dates.extend(reg1)
dates.extend(reg2)
dates.extend(reg3)
dates.extend(reg4)
dates.extend(reg5)

dates = list(set(dates))
```

```
In [27]: dates[:10]

Out[27]: ['April 22, 1993',
          '1993Apr11',
          '04/01/93',
          '1993Apr15',
          'March 4, 1993',
          '1993Apr30',
          '1993Mar27',
          'December 31, 1992',
          '1993Apr16',
          'October 15, 1976']

In [ ]:
```