In [5]:

```
import pandas as pd
import nltk
text="""\
In the year 1878 I took my degree of Doctor of Medicine of the University of London, and proceeded to Netley to go through the course prescri
bed for surgeons in the army. Having completed my studies there, I was duly attached to the Fifth Northumberland Fusiliers as Assistant Surge
on. The regiment was stationed in India at the time, and before I could join it, the second Afghan war had broken out. On landing at Bombay,
 I learned that my corps had advanced through the passes, and was already deep in the enemy's country. I followed, however, with many other o
fficers who were in the same situation as myself, and succeeded in reaching Candahar in safety, where I found my regiment, and at once entere
d upon my new duties.

The campaign brought honours and promotion to many, but for me it had nothing but misfortune and disaster. I was removed from my brigade and
 attached to the Berkshires, with whom I served at the fatal battle of Maiwand. There I was struck on the shoulder by a Jezail bullet, which
 shattered the bone and grazed the subclavian artery. I should have fallen into the hands of the murderous Ghazis had it not been for the dev
otion and courage shown by Murray, my orderly, who threw me across a pack-horse, and succeeded in bringing me safely to the British lines.
"""
f = open('stud.txt')
longtext = f.read()
f.close()
```

Segmentation (Sentence)

In [6]:

```
s=nltk.sent_tokenize(text)
print("\n--\n".join(s))
```

```
In the year 1878 I took my degree of Doctor of Medicine of the University of London, and proceeded to Netley to go through the course prescr
ibed for surgeons in the army.
--
Having completed my studies there, I was duly attached to the Fifth Northumberland Fusiliers as Assistant Surgeon.
--
The regiment was stationed in India at the time, and before I could join it, the second Afghan war had broken out.
--
On landing at Bombay, I learned that my corps had advanced through the passes, and was already deep in the enemy's country.
--
I followed, however, with many other officers who were in the same situation as myself, and succeeded in reaching Candahar in safety, where
I found my regiment, and at once entered upon my new duties.
--
The campaign brought honours and promotion to many, but for me it had nothing but misfortune and disaster.
--
I was removed from my brigade and attached to the Berkshires, with whom I served at the fatal battle of Maiwand.
--
There I was struck on the shoulder by a Jezail bullet, which shattered the bone and grazed the subclavian artery.
--
I should have fallen into the hands of the murderous Ghazis had it not been for the devotion and courage shown by Murray, my orderly, who th
rew me across a pack-horse, and succeeded in bringing me safely to the British lines.
```

In [7]:

```
s=nltk.sent_tokenize("Dr. Weber will help you. We go to the U.S.A., i.e., not to Italy; Mrs. Hussey is there!")
print("\n--\n".join(s))
```

```
Dr. Weber will help you.
--
We go to the U.S.A., i.e., not to Italy; Mrs. Hussey is there!
```

In [8]:

```
s=nltk.sent_tokenize('''"A clam for supper? a cold clam; is THAT what you mean, Mrs. Hussey?" says I, "but that's a rather cold and clammy reception in the winter time, ain't it, Mrs. Hussey?"''')
print("\n--\n".join(s))
```

```
"A clam for supper?
--
a cold clam; is THAT what you mean, Mrs.
--
Hussey?"
--
says I, "but that's a rather cold and clammy reception in the winter time, ain't it, Mrs.
--
Hussey?"
```

Word Tokenization

```
In [9]: tokens=nltk.word_tokenize(text)

        pd.DataFrame(data=tokens[0:30],columns=["token"])
```

`Out[9]:`

	token
0	In
1	the
2	year
3	1878
4	I
5	took
6	my
7	degree
8	of
9	Doctor
10	of
11	Medicine
12	of
13	the
14	University
15	of
16	London
17	,
18	and
19	proceeded
20	to
21	Netley
22	to
23	go
24	through
25	the
26	course
27	prescribed
28	for
29	surgeons

```
In [10]: tokens=nltk.word_tokenize("20% hundreds dog's house (here) and 'there he goes' and \"somwhere he flows\"")

         pd.DataFrame(data=tokens,columns=["token"])
```

Out[10]:

	token
0	20
1	%
2	hundreds
3	dog
4	's
5	house
6	(
7	here
8	)
9	and
10	'there
11	he
12	goes
13	'
14	and
15	``
16	somwhere
17	he
18	flows
19	''

bigram - only frequencies

In [11]:

```
tokens=nltk.word_tokenize(longtext)
words=[word.lower() for word in tokens if word.isalpha()]

bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = nltk.collocations.BigramCollocationFinder.from_words(words)
result=finder.score_ngrams(bigram_measures.raw_freq)[0:20]

print("%-10s\t%-10s\t%s" % ("term 1", "term 2", "freq"))
print("----------------------------------------")
print("\n".join(list(str("%-10s\t%-10s\t%.6f" % (bigram[0],bigram[1],freq)) for bigram, freq in result)))
```

```
term 1          term 2          freq
----------------------------------------
of              the             0.006964
in              the             0.004875
to              the             0.003180
to              be              0.002275
he              had             0.002182
it              was             0.002182
and             the             0.002089
upon            the             0.002043
at              the             0.001973
he              was             0.001927
i               have            0.001880
that            i               0.001764
that            he              0.001718
he              said            0.001695
on              the             0.001695
there           was             0.001695
of              his             0.001602
from            the             0.001509
had             been            0.001486
of              a               0.001462
```

bigram - pmi without frequency threshold

In [12]:
```
result=finder.score_ngrams(bigram_measures.pmi)[0:20]

print("%-10s\t%-10s\t%s" % ("term 1", "term 2", "score"))
print("--------------------------------------")
print("\n".join(list(str("%-10s\t%-10s\t%.2f" % (bigram[0],bigram[1],score)) for bigram, score in result)))
```

```
term 1          term 2          score
-------------------------------------
ac              nummos          15.39
admired         treated         15.39
airy            cheerfully      15.39
ambitious       title           15.39
anchor          tattooed        15.39
angel           merona          15.39
aqua            tofana          15.39
arch            rebel           15.39
assistant       surgeon         15.39
attractive      locality        15.39
audible         expressions     15.39
babe            unborn          15.39
balsamic        odour           15.39
barrenness      inhospitality   15.39
basaltic        columns         15.39
belladonna      opium           15.39
big             pitcher         15.39
blanched        skeletons       15.39
bodily          exertion        15.39
brain           originally      15.39
```

bigram - pmi with frequency threshold

In [13]:

```python
finder.apply_freq_filter(10)
result=finder.score_ngrams(bigram_measures.pmi)[0:20]

print("%-20s\t%s\t%s\t%s\t%s" % ("bigram", "tf 1", "tf 2", "tf 1&2","score"))
print("-----------------------------------------------------")
print("\n".join(list(str("%-20s\t%d\t%d\t%d\t%.2f" % (" ".join(bigram),finder.word_fd[bigram[0]],finder.word_fd[bigram[1]],finder.ngram_fd[bigram],score)) for bigram, score in result)))
```

```
bigram                  tf 1    tf 2    tf 1&2  score
-----------------------------------------------------
salt lake               11      10      10      11.94
brixton road            15      28      13      10.38
jefferson hope          37      56      34      9.47
joseph stangerson       13      47      10      9.46
john ferrier            39      62      29      9.01
sherlock holmes         52      98      52      8.78
lucy ferrier            29      62      10      7.90
no doubt                174     19      17      7.79
more than               83      57      19      7.43
her father              173     28      12      6.74
my companion            306     41      29      6.64
at last                 319     49      22      5.92
my own                  306     51      19      5.71
did not                 59      185     13      5.68
they were               180     169     35      5.63
has been                80      147     13      5.57
at once                 319     37      13      5.57
may be                  45      250     12      5.52
will be                 94      250     25      5.52
might have              43      290     13      5.49
```

https://books.google.com/ngrams (https://books.google.com/ngrams)

In [49]:

```
tokens=nltk.word_tokenize(longtext)
words=[word.lower() for word in tokens if word.isalpha()]

trigram_measures = nltk.collocations.TrigramAssocMeasures()
finder = nltk.collocations.TrigramCollocationFinder.from_words(words)
finder.apply_freq_filter(5)
result=finder.score_ngrams(trigram_measures.pmi)[0:20]

print("%-30s\t%s\t%s\t%s\t%s\t%s" % ("bigram", "tf 1", "tf 2", "tf 3", "tf1&2&3","score"))
print("-----------------------------------------------------------------------")
print("\n".join(list(str("%-30s\t%d\t%d\t%d\t%d\t%.2f" % (" ".join(trigram),finder.word_fd[trigram[0]],finder.word_fd[trigram[2]],finder.word
_fd[trigram[2]],finder.ngram_fd[trigram],score)) for trigram, score in result)))
```

```
bigram                          tf 1    tf 2    tf 3    tf1&2&3 score
-----------------------------------------------------------------------
halliday private hotel          5       14      14      5       23.40
salt lake city                  11      23      23      9       22.65
the brixton road                2535    28      28      11      14.23
of enoch drebber                1217    62      62      5       13.91
of joseph stangerson            1217    47      47      5       13.61
said sherlock holmes            207     98      98      7       13.59
may as well                     45      58      58      5       13.38
do not know                     125     50      50      6       13.23
jefferson hope was              37      653     653     5       12.74
joseph stangerson the           13      2535    2535    5       12.55
i should like                   927     34      34      5       12.33
the two detectives              2535    9       9       5       12.33
the young hunter                2535    14      14      5       12.30
must have been                  44      147     147     5       12.27
should like to                  57      1088    1088    5       12.10
no doubt that                   174     673     673     5       12.03
as far as                       333     333     333     6       11.97
as he spoke                     333     28      28      15      11.86
said at last                    207     49      49      6       11.75
gregson and lestrade            46      47      47      5       11.63
```

Word Tagging

In [14]:

```
tokens=nltk.word_tokenize(text)
tags=nltk.pos_tag(tokens)[0:20]

pd.DataFrame(data=tags, columns=["term","tag"])
```

Out[14]:

	term	tag
0	In	IN
1	the	DT
2	year	NN
3	1878	CD
4	I	PRP
5	took	VBD
6	my	PRP$
7	degree	NN
8	of	IN
9	Doctor	NNP
10	of	IN
11	Medicine	NNP
12	of	IN
13	the	DT
14	University	NNP
15	of	IN
16	London	NNP
17	,	,
18	and	CC
19	proceeded	VBD

In [15]:

```
tokens=nltk.word_tokenize(text)
tags=nltk.pos_tag(tokens, tagset="universal")[0:20]

pd.DataFrame(data=tags, columns=["term","tag"])
```

Out[15]:

	term	tag
0	In	ADP
1	the	DET
2	year	NOUN
3	1878	NUM
4	I	PRON
5	took	VERB
6	my	PRON
7	degree	NOUN
8	of	ADP
9	Doctor	NOUN
10	of	ADP
11	Medicine	NOUN
12	of	ADP
13	the	DET
14	University	NOUN
15	of	ADP
16	London	NOUN
17	,	.
18	and	CONJ
19	proceeded	VERB

In [16]:

```
tokens=nltk.word_tokenize(longtext)
result=nltk.FreqDist(tag for (word, tag) in nltk.pos_tag(tokens)).most_common()

pd.DataFrame(data=result, columns=["tag","freq"])
```

Out[16]:

	tag	freq
0	NN	6170
1	IN	5572
2	DT	4692
3	PRP	4067
4	VBD	3387
5	,	2959
6	.	2699
7	JJ	2648
8	RB	2255
9	CC	1711
10	NNP	1668
11	VB	1555
12	NNS	1498
13	PRP$	1369
14	VBN	1242
15	TO	1088
16	"	918
17	``	886
18	VBP	745
19	VBG	713
20	MD	658
21	VBZ	630
22	CD	354
23	WDT	349
24	RP	285
25	WRB	271
26	WP	258
27	:	247
28	EX	202
29	POS	162

	tag	freq
30	JJR	100
31	PDT	64
32	JJS	64
33	RBR	51
34	UH	30
35	RBS	30
36	NNPS	22
37	WP$	11
38	FW	6
39	(	2
40	)	2

In [17]:

```
tokens=nltk.word_tokenize(longtext)
result=nltk.FreqDist(tag for (word, tag) in nltk.pos_tag(tokens, tagset="universal")).most_common()

pd.DataFrame(data=result, columns=["tag","freq"])
```

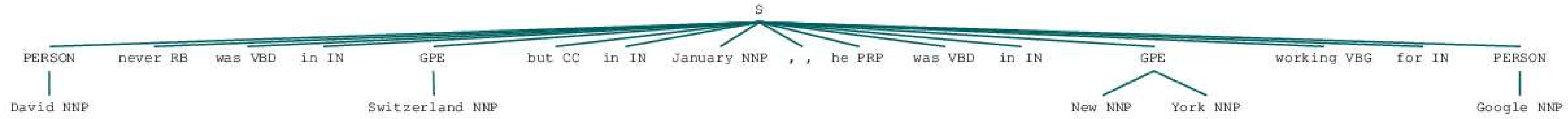
Out[17]:

	tag	freq
0	NOUN	9358
1	VERB	8930
2	.	7713
3	PRON	5705
4	ADP	5572
5	DET	5307
6	ADJ	2812
7	ADV	2607
8	CONJ	1711
9	PRT	1535
10	NUM	354
11	X	36

In [18]:

```
# extract named entities (ne_chunk)
pos=nltk.pos_tag(nltk.word_tokenize("David never was in Switzerland but in January, he was in New York working for Google"))
nltk.ne_chunk(pos)
```

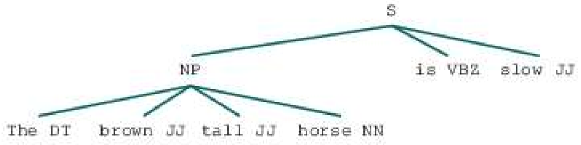
Out[18]:



In [27]:

```
# extract noun phrases (NP)
grammar = "NP: {<DT>?<JJ>*<NN>}"
pos=nltk.pos_tag(nltk.word_tokenize("The brown tall horse is slow"))
nltk.RegexpParser(grammar).parse(pos)
```

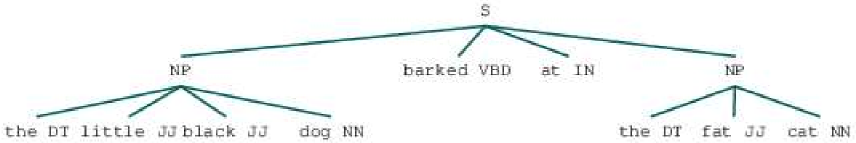
Out[27]:



In [28]:

```
pos=nltk.pos_tag(nltk.word_tokenize("the little black dog barked at the fat cat"))
nltk.RegexpParser(grammar).parse(pos)
```

Out[28]:



In [54]:

```
nltk.pos_tag(nltk.word_tokenize("He wanted to ski. He found the ski. Can I ski?"))
```

Out[54]:

```
[('He', 'PRP'),
 ('wanted', 'VBD'),
 ('to', 'TO'),
 ('ski', 'VB'),
 ('.', '.'),
 ('He', 'PRP'),
 ('found', 'VBD'),
 ('the', 'DT'),
 ('ski', 'NN'),
 ('.', '.'),
 ('Can', 'MD'),
 ('I', 'PRP'),
 ('ski', 'VB'),
 ('?', '.')]
```

Lemmatization

In [55]:

```
from nltk.corpus import wordnet as wn

def is_noun(tag):
    return tag in ['NN', 'NNS', 'NNP', 'NNPS']


def is_verb(tag):
    return tag in ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']


def is_adverb(tag):
    return tag in ['RB', 'RBR', 'RBS']


def is_adjective(tag):
    return tag in ['JJ', 'JJR', 'JJS']


def penn_to_wn(tag):
    if is_adjective(tag):
        return wn.ADJ
    elif is_noun(tag):
        return wn.NOUN
    elif is_adverb(tag):
        return wn.ADV
    elif is_verb(tag):
        return wn.VERB
    return wn.ADJ
```

In [56]:

```
pos=nltk.pos_tag(nltk.word_tokenize(text))
porter=nltk.PorterStemmer()
lancaster=nltk.LancasterStemmer()
snowball = nltk.SnowballStemmer("english")
wordnet=nltk.WordNetLemmatizer()
stems=list((w.lower(),penn_to_wn(p),porter.stem(w).lower(),lancaster.stem(w).lower(),snowball.stem(w).lower(),wordnet.lemmatize(w,pos=penn_to_wn(p)).lower()) for w,p in pos)

print("%-15s %-15s %-15s %-15s %-15s %-15s" % ("Term", "pos tag","Porter","Lancaster","Snowball","Wordnet"))
print("--------------------------------------------------------------------------------------------------")
print("\n".join(list(str("%-15s %-15s %-15s %-15s %-15s %-15s" % (t,pos,p,l,s,w)) for t,pos,p,l,s,w in stems[0:40] if w!=t or t!=p or p!=l or l!=s or s!=w)))
```

```
Term            pos tag         Porter          Lancaster       Snowball        Wordnet
--------------------------------------------------------------------------------------------------
took            v               took            took            took            take
degree          n               degre           degr            degre           degree
doctor          n               doctor          doct            doctor          doctor
medicine        n               medicin         medicin         medicin         medicine
university      n               univers         univers         univers         university
proceeded       v               proceed         process         proceed         proceed
course          n               cours           cours           cours           course
prescribed      v               prescrib        prescrib        prescrib        prescribe
surgeons        n               surgeon         surgeon         surgeon         surgeon
army            n               armi            army            armi            army
having          v               have            hav             have            having
completed       v               complet         complet         complet         complete
studies         n               studi           study           studi           study
there           r               there           ther            there           there
```

In [57]:

```python
nltk.corpus.wordnet.synsets('dog')

print("defintions:\n-------------------------------------------")
print("\n".join(list(str("%-20s %s" % (s.name(),s.definition())) for s in nltk.corpus.wordnet.synsets('dog'))))
print("\n\nexamples:\n-------------------------------------------")
print("\n".join(list(str("%-20s %s" % (s.name(),s.examples())) for s in nltk.corpus.wordnet.synsets('dog'))))
print("\n\nlemma names:\n-------------------------------------------")
print("\n".join(list(str("%-20s %s" % (s.name(),s.lemma_names())) for s in nltk.corpus.wordnet.synsets('dog'))))
```

```
defintions:
-------------------------------------------
dog.n.01             a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistor
ic times; occurs in many breeds
frump.n.01           a dull unattractive unpleasant girl or woman
dog.n.03             informal term for a man
cad.n.01             someone who is morally reprehensible
frank.n.02           a smooth-textured sausage of minced beef or pork usually smoked; often served on a bread roll
pawl.n.01            a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward
andiron.n.01         metal supports for logs in a fireplace
chase.v.01           go after with the intent to catch


examples:
-------------------------------------------
dog.n.01             ['the dog barked all night']
frump.n.01           ['she got a reputation as a frump', "she's a real dog"]
dog.n.03             ['you lucky dog']
cad.n.01             ['you dirty dog']
frank.n.02           []
pawl.n.01            []
andiron.n.01         ['the andirons were too hot to touch']
chase.v.01           ['The policeman chased the mugger down the alley', 'the dog chased the rabbit']


lemma names:
-------------------------------------------
dog.n.01             ['dog', 'domestic_dog', 'Canis_familiaris']
frump.n.01           ['frump', 'dog']
dog.n.03             ['dog']
cad.n.01             ['cad', 'bounder', 'blackguard', 'dog', 'hound', 'heel']
frank.n.02           ['frank', 'frankfurter', 'hotdog', 'hot_dog', 'dog', 'wiener', 'wienerwurst', 'weenie']
pawl.n.01            ['pawl', 'detent', 'click', 'dog']
andiron.n.01         ['andiron', 'firedog', 'dog', 'dog-iron']
chase.v.01           ['chase', 'chase_after', 'trail', 'tail', 'tag', 'give_chase', 'dog', 'go_after', 'track']
```

In [58]:

```
nltk.corpus.wordnet.synsets('horse')

print("hypernyms:\n-------------------------------------------")
print("\n".join(list(str("%-20s %s" % (s.name(),s.hypernyms())) for s in nltk.corpus.wordnet.synsets('horse'))))
print("\n\nhyponyms:\n-------------------------------------------")
print("\n".join(list(str("%-20s %s" % (s.name(),s.hyponyms())) for s in nltk.corpus.wordnet.synsets('horse'))))
```

```
hypernyms:
-------------------------------------------
horse.n.01           [Synset('equine.n.01')]
horse.n.02           [Synset('gymnastic_apparatus.n.01')]
cavalry.n.01         [Synset('military_personnel.n.01')]
sawhorse.n.01        [Synset('framework.n.03')]
knight.n.02          [Synset('chessman.n.01')]
horse.v.01           [Synset('provide.v.02')]


hyponyms:
-------------------------------------------
horse.n.01           [Synset('bay.n.07'), Synset('chestnut.n.06'), Synset('eohippus.n.01'), Synset('gee-gee.n.01'), Synset('hack.n.06'), Syn
set('hack.n.07'), Synset('harness_horse.n.01'), Synset('liver_chestnut.n.01'), Synset('male_horse.n.01'), Synset('mare.n.01'), Synset('mesoh
ippus.n.01'), Synset('pacer.n.02'), Synset('palomino.n.01'), Synset('pinto.n.01'), Synset('polo_pony.n.01'), Synset('pony.n.01'), Synset('po
ny.n.05'), Synset('post_horse.n.01'), Synset('protohippus.n.01'), Synset('racehorse.n.01'), Synset('roan.n.02'), Synset('saddle_horse.n.0
1'), Synset('sorrel.n.05'), Synset('stablemate.n.01'), Synset('stalking-horse.n.04'), Synset('steeplechaser.n.01'), Synset('stepper.n.03'),
Synset('wild_horse.n.01'), Synset('workhorse.n.02')]
horse.n.02           [Synset('pommel_horse.n.01'), Synset('vaulting_horse.n.01')]
cavalry.n.01         []
sawhorse.n.01        [Synset('trestle.n.02')]
knight.n.02          []
horse.v.01           [Synset('remount.v.03')]
```

Aggregation

In [ ]:

```
nltk.corpus.stopwords.words('english')
#nltk.corpus.stopwords.words('german')
#nltk.corpus.stopwords.words('italian')
#nltk.corpus.stopwords.words()
```

In [59]:

```
stopwords=nltk.corpus.stopwords.words('english')
tokens=nltk.word_tokenize(longtext)
for w in nltk.FreqDist(tokens).most_common(20):
        print("%-20s %s" %w)

print("\n\nstopwords: %.3f" % (len([w for w in tokens if w.lower() in stopwords])/len(tokens)))
print("non-alpha: %.3f" % (len([w for w in tokens if not w.isalpha()])/len(tokens)))
print("  content: %.3f" % (len([w for w in tokens if w.isalpha() and w.lower() not in stopwords])/len(tokens)))
```

```
,                    2959
.                    2406
the                  2327
and                  1322
of                   1204
to                   1076
a                    963
I                    927
``                   886
''                   811
in                   674
was                  649
he                   630
that                 619
his                  613
had                  471
it                   453
you                  369
which                315
with                 313


stopwords: 0.458
non-alpha: 0.166
  content: 0.376
```