In [1]:
```python
import nltk
```

In [2]:
```python
nltk.download("movie_reviews")
```
```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\Hadi\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\movie_reviews.zip.
```
Out[2]: `True`

In [25]:
```python
from nltk.corpus import movie_reviews
```

In [ ]:
```python
nltk.download()
```
```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

In [26]:
```python
from nltk.corpus import movie_reviews
```

In [27]:
```python
len(movie_reviews.fileids())
```
Out[27]: `2000`

In [28]:
```python
movie_reviews.fileids()[:5]
```
Out[28]:
```
['neg/cv000_29416.txt',
 'neg/cv001_19502.txt',
 'neg/cv002_17424.txt',
 'neg/cv003_12683.txt',
 'neg/cv004_12641.txt']
```

In [29]:
```python
movie_reviews.fileids()[-5:]
```
Out[29]:
```
['pos/cv995_21821.txt',
 'pos/cv996_11592.txt',
 'pos/cv997_5046.txt',
 'pos/cv998_14111.txt',
 'pos/cv999_13106.txt']
```

In [30]:
```python
negative_fileids = movie_reviews.fileids('neg')
positive_fileids = movie_reviews.fileids('pos')
```

In [31]:
```python
len(negative_fileids), len(positive_fileids)
```
Out[31]: `(1000, 1000)`

In [32]:
```python
print(movie_reviews.raw(fileids=positive_fileids[0]))
```
```
films adapted from comic books have had plenty of success , whether they're about superheroes
( batman , superman , spawn ) , or geared toward kids ( casper ) or the arthouse crowd ( ghos
t world ) , but there's never really been a comic book like from hell before .
for starters , it was created by alan moore ( and eddie campbell ) , who brought the medium t
o a whole new level in the mid '80s with a 12-part series called the watchmen .
to say moore and campbell thoroughly researched the subject of jack the ripper would be like
saying michael jackson is starting to look a little odd .
the book ( or " graphic novel , " if you will ) is over 500 pages long and includes nearly 30
more that consist of nothing but footnotes .
in other words , don't dismiss this film because of its source .
if you can get past the whole comic book thing , you might find another stumbling block in fr
om hell's directors , albert and allen hughes .
getting the hughes brothers to direct this seems almost as ludicrous as casting carrot top in
, well , anything , but riddle me this : who better to direct a film that's set in the ghetto
and features really violent street crime than the mad geniuses behind menace ii society ?
the ghetto in question is , of course , whitechapel in 1888 london's east end .
it's a filthy , sooty place where the whores ( called " unfortunates " ) are starting to get
a little nervous about this mysterious psychopath who has been carving through their professi
on with surgical precision .
when the first stiff turns up , copper peter godley ( robbie coltrane , the world is not enou
gh ) calls in inspector frederick abberline ( johnny depp , blow ) to crack the case .
abberline , a widower , has prophetic dreams he unsuccessfully tries to quell with copious am
ounts of absinthe and opium .
upon arriving in whitechapel , he befriends an unfortunate named mary kelly ( heather graham
, say it isn't so ) and proceeds to investigate the horribly gruesome crimes that even the po
lice surgeon can't stomach .
i don't think anyone needs to be briefed on jack the ripper , so i won't go into the particul
ars here , other than to say moore and campbell have a unique and interesting theory about bo
th the identity of the killer and the reasons he chooses to slay .
in the comic , they don't bother cloaking the identity of the ripper , but screenwriters terr
y hayes ( vertical limit ) and rafael yglesias ( les mis ? rables ) do a good job of keeping
him hidden from viewers until the very end .
it's funny to watch the locals blindly point the finger of blame at jews and indians because
, after all , an englishman could never be capable of committing such ghastly acts .
and from hell's ending had me whistling the stonecutters song from the simpsons for days ( "
who holds back the electric car/who made steve guttenberg a star ? " ) .
don't worry - it'll all make sense when you see it .
now onto from hell's appearance : it's certainly dark and bleak enough , and it's surprising
to see how much more it looks like a tim burton film than planet of the apes did ( at times ,
it seems like sleepy hollow 2 ) .
the print i saw wasn't completely finished ( both color and music had not been finalized , so
no comments about marilyn manson ) , but cinematographer peter deming ( don't say a word ) ab
ly captures the dreariness of victorian-era london and helped make the flashy killing scenes
remind me of the crazy flashbacks in twin peaks , even though the violence in the film pales
in comparison to that in the black-and-white comic .
oscar winner martin childs' ( shakespeare in love ) production design turns the original prag
ue surroundings into one creepy place .
even the acting in from hell is solid , with the dreamy depp turning in a typically strong pe
rformance and deftly handling a british accent .
ians holm ( joe gould's secret ) and richardson ( 102 dalmatians ) log in great supporting ro
les , but the big surprise here is graham .
i cringed the first time she opened her mouth , imagining her attempt at an irish accent , bu
t it actually wasn't half bad .
the film , however , is all good .
2 : 00 - r for strong violence/gore , sexuality , language and drug content
```

Tokenize Text in Words

In [4]:
```python
romeo_text = """Why then, O brawling love! O loving hate!
O any thing, of nothing first create!
O heavy lightness, serious vanity,
Misshapen chaos of well-seeming forms,
Feather of lead, bright smoke, cold fire, sick health,
Still-waking sleep, that is not what it is!
This love feel I, that feel no love in this."""
```

In [5]:
```python
romeo_text.split()
```
Out[5]:
```
['Why',
 'then,',
 'O',
 'brawling',
 'love!',
 'O',
 'loving',
 'hate!',
 'O',
 'any',
 'thing,',
 'of',
 'nothing',
 'first',
 'create!',
 'O',
 'heavy',
 'lightness,',
 'serious',
 'vanity,',
 'Misshapen',
 'chaos',
 'of',
 'well-seeming',
 'forms,',
 'Feather',
 'of',
 'lead,',
 'bright',
 'smoke,',
 'cold',
 'fire,',
 'sick',
 'health,',
 'Still-waking',
 'sleep,',
 'that',
 'is',
 'not',
 'what',
 'it',
 'is!',
 'This',
 'love',
 'feel',
 'I,',
 'that',
 'feel',
 'no',
 'love',
 'in',
 'this.']
```

In [6]:
```python
nltk.download("punkt")
```
```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Hadi\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```
Out[6]: `True`

In [7]:
```python
romeo_words = nltk.word_tokenize(romeo_text)
```

In [8]:
```python
romeo_words
```
Out[8]:
```
['Why',
 'then',
 ',',
 'O',
 'brawling',
 'love',
 '!',
 'O',
 'loving',
 'hate',
 '!',
 'O',
 'any',
 'thing',
 ',',
 'of',
 'nothing',
 'first',
 'create',
 '!',
 'O',
 'heavy',
 'lightness',
 ',',
 'serious',
 'vanity',
 ',',
 'Misshapen',
 'chaos',
 'of',
 'well-seeming',
 'forms',
 ',',
 'Feather',
 'of',
 'lead',
 ',',
 'bright',
 'smoke',
 ',',
 'cold',
 'fire',
 ',',
 'sick',
 'health',
 ',',
 'Still-waking',
 'sleep',
 ',',
 'that',
 'is',
 'not',
 'what',
 'it',
 'is',
 '!',
 'This',
 'love',
 'feel',
 'I',
 ',',
 'that',
 'feel',
 'no',
 'love',
 'in',
 'this',
 '.']
```

In [55]:
```python
movie_reviews.words(fileids=positive_fileids[0])
```
Out[55]: `['films', 'adapted', 'from', 'comic', 'books', 'have', ...]`

Build a bag-of-words model

In [10]:
```python
{word:True for word in romeo_words}
```
Out[10]:
```
{'Why': True,
 'then': True,
 ',': True,
 'O': True,
 'brawling': True,
 'love': True,
 '!': True,
 'loving': True,
 'hate': True,
 'any': True,
 'thing': True,
 'of': True,
 'nothing': True,
 'first': True,
 'create': True,
 'heavy': True,
 'lightness': True,
 'serious': True,
 'vanity': True,
 'Misshapen': True,
 'chaos': True,
 'well-seeming': True,
 'forms': True,
 'Feather': True,
 'lead': True,
 'bright': True,
 'smoke': True,
 'cold': True,
 'fire': True,
 'sick': True,
 'health': True,
 'Still-waking': True,
 'sleep': True,
 'that': True,
 'is': True,
 'not': True,
 'what': True,
 'it': True,
 'This': True,
 'feel': True,
 'I': True,
 'no': True,
 'in': True,
 'this': True,
 '.': True}
```

In [11]:
```python
type(_)
```
Out[11]: `dict`

In [12]:
```python
def build_bag_of_words_features(words):
    return {word:True for word in words}
```

In [13]:
```python
build_bag_of_words_features(romeo_words)
```
Out[13]:
```
{'Why': True,
 'then': True,
 ',': True,
 'O': True,
 'brawling': True,
 'love': True,
 '!': True,
 'loving': True,
 'hate': True,
 'any': True,
 'thing': True,
 'of': True,
 'nothing': True,
 'first': True,
 'create': True,
 'heavy': True,
 'lightness': True,
 'serious': True,
 'vanity': True,
 'Misshapen': True,
 'chaos': True,
 'well-seeming': True,
 'forms': True,
 'Feather': True,
 'lead': True,
 'bright': True,
 'smoke': True,
 'cold': True,
 'fire': True,
 'sick': True,
 'health': True,
 'Still-waking': True,
 'sleep': True,
 'that': True,
 'is': True,
 'not': True,
 'what': True,
 'it': True,
 'This': True,
 'feel': True,
 'I': True,
 'no': True,
 'in': True,
 'this': True,
 '.': True}
```

In [14]:
```python
nltk.download("stopwords")
```
```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Hadi\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```
Out[14]: `True`

In [15]:
```python
import string
import data
```

In [16]:
```python
string.punctuation
```
Out[16]: `'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'`

In [22]:
```python
useless_words = nltk.corpus.stopwords.words("english") + list(string.punctuation)
useless_words
type(useless_words)
```
Out[22]: `list`

In [23]:
```python
def build_bag_of_words_features_filtered(words):
    return {
        word:1 for word in words \
        if not word in useless_words}
```

In [24]:
```python
build_bag_of_words_features_filtered(romeo_words)
```
Out[24]:
```
{'Why': 1,
 'O': 1,
 'brawling': 1,
 'love': 1,
 'loving': 1,
 'hate': 1,
 'thing': 1,
 'nothing': 1,
 'first': 1,
 'create': 1,
 'heavy': 1,
 'lightness': 1,
 'serious': 1,
 'vanity': 1,
 'Misshapen': 1,
 'chaos': 1,
 'well-seeming': 1,
 'forms': 1,
 'Feather': 1,
 'lead': 1,
 'bright': 1,
 'smoke': 1,
 'cold': 1,
 'fire': 1,
 'sick': 1,
 'health': 1,
 'Still-waking': 1,
 'sleep': 1,
 'This': 1,
 'feel': 1,
 'I': 1}
```

a flatter distribuation indicates a large vocabulary , while a peak distribuation indicated a restricted vocabulary. often due to a focused topic ore specialize language.

Plotting Frequencies of Words

In [33]:
```python
all_words = movie_reviews.words()
len(all_words)/1e6
```
Out[33]: `1.58382`

In [34]:
```python
filtered_words = [word for word in movie_reviews.words() if not word in useless_words]
type(filtered_words)
```
Out[34]: `list`

In [35]:
```python
len(filtered_words)/1e6
```
Out[35]: `0.710579`

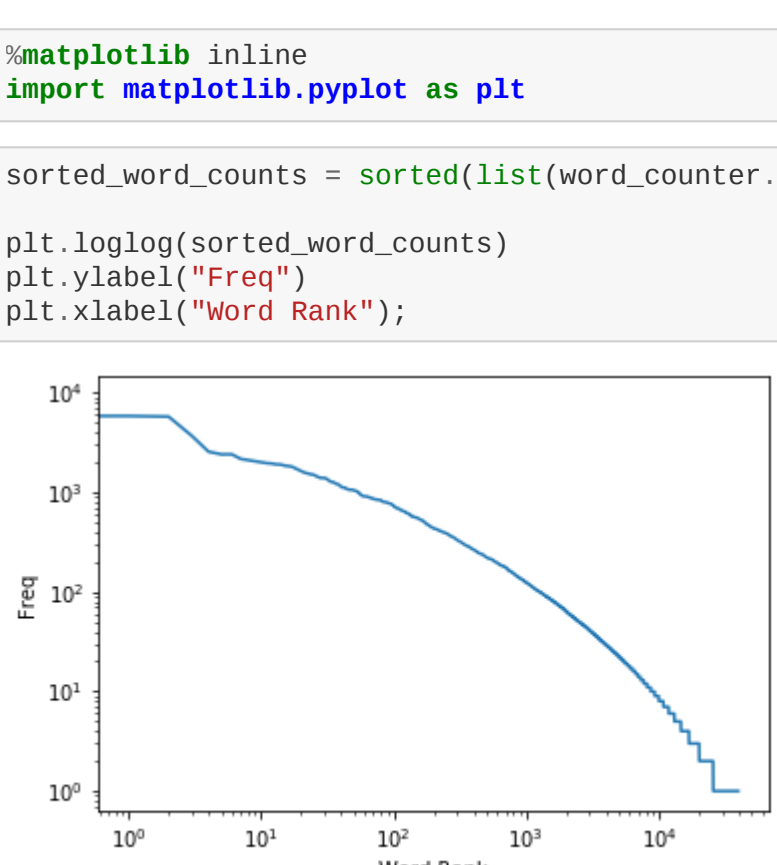
In [36]:
```python
from collections import Counter

word_counter = Counter(filtered_words)
```

In [37]:
```python
most_common_words = word_counter.most_common()[:10]
```

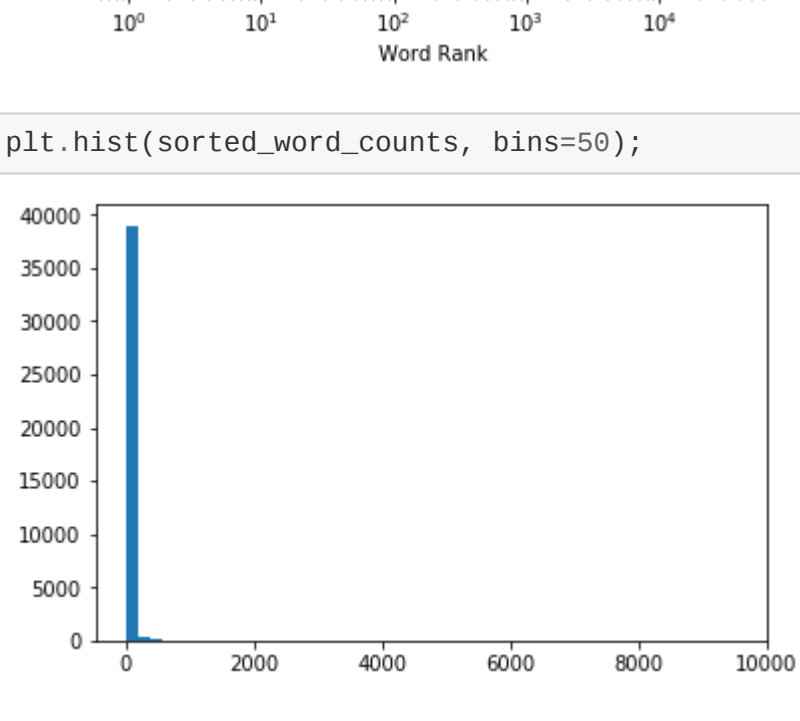
In [38]:
```python
most_common_words
```
Out[38]:
```
[('film', 9517),
 ('one', 5852),
 ('movie', 5771),
 ('like', 3690),
 ('even', 2565),
 ('good', 2411),
 ('time', 2411),
 ('story', 2169),
 ('would', 2109),
 ('much', 2049)]
```

In [39]:
```python
%matplotlib inline
import matplotlib.pyplot as plt
```

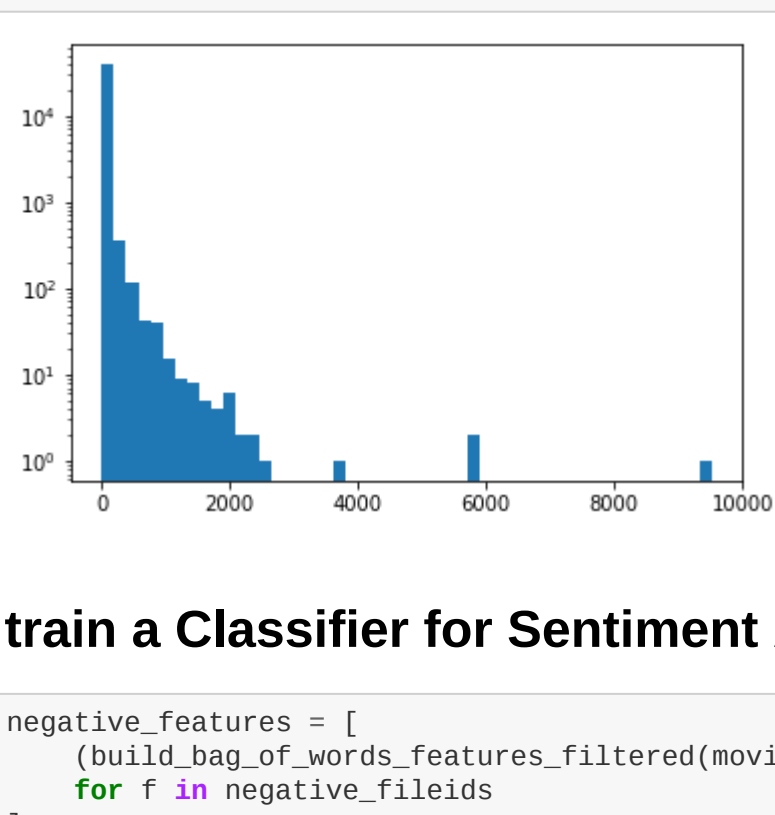
In [40]:
```python
sorted_word_counts = sorted(list(word_counter.values()), reverse=True)

plt.loglog(sorted_word_counts)
plt.ylabel("Freq")
plt.xlabel("Word Rank");
```



In [41]:
```python
plt.hist(sorted_word_counts, bins=50);
```



In [42]:
```python
plt.hist(sorted_word_counts, bins=50, log=True);
```



train a Classifier for Sentiment Analysis

In [45]:
```python
negative_features = [
    (build_bag_of_words_features_filtered(movie_reviews.words(fileids=[f])), 'neg') \
    for f in negative_fileids
]
```

In [46]:
```python
print(negative_features[3])
```
```
({'quest': 1, 'camelot': 1, 'warner': 1, 'bros': 1, 'first': 1, 'feature': 1, 'length': 1, 'f
ully': 1, 'animated': 1, 'attempt': 1, 'steal': 1, 'clout': 1, 'disney': 1, 'cartoon': 1, 'em
pire': 1, 'mouse': 1, 'reason': 1, 'worried': 1, 'recent': 1, 'challenger': 1, 'throne': 1,
'last': 1, 'fall': 1, 'promising': 1, 'flawed': 1, '20th': 1, 'century': 1, 'fox': 1, 'produc
tion': 1, 'anastasia': 1, 'hercules': 1, 'lively': 1, 'cast': 1, 'colorful': 1, 'palate': 1,
'beat': 1, 'hands': 1, 'came': 1, 'time': 1, 'crown': 1, '1997': 1, 'best': 1, 'piece': 1, 'a
nimation': 1, 'year': 1, 'contest': 1, 'pretty': 1, 'much': 1, 'dead': 1, 'arrival': 1, 'eve
n': 1, 'magic': 1, 'kingdom': 1, 'mediocre': 1, '--': 1, 'pocahontas': 1, 'keeping': 1, 'scor
e': 1, 'nearly': 1, 'dull': 1, 'story': 1, 'revolves': 1, 'around': 1, 'adventures': 1, 'fre
e': 1, 'spirited': 1, 'kayley': 1, 'voiced': 1, 'jessalyn': 1, 'gilsig': 1, 'early': 1, 'tee
n': 1, 'daughter': 1, 'belated': 1, 'knight': 1, 'king': 1, 'arthur': 1, 'round': 1, 'table':
1, 'dream': 1, 'follow': 1, 'father': 1, 'footsteps': 1, 'gets': 1, 'chance': 1, 'evil': 1,
'warlord': 1, 'ruber': 1, 'gary': 1, 'oldman': 1, 'ex': 1, 'member': 1, 'gone': 1, 'bad': 1,
'steals': 1, 'magical': 1, 'sword': 1, 'excalibur': 1, 'accidentally': 1, 'loses': 1, 'danger
ous': 1, 'booby': 1, 'trapped': 1, 'forest': 1, 'help': 1, 'hunky': 1, 'blind': 1, 'timberlan
d': 1, 'dweller': 1, 'garrett': 1, 'carey': 1, 'elwes': 1, 'two': 1, 'headed': 1, 'dragon':
1, 'eric': 1, 'idle': 1, 'rickles': 1, 'always': 1, 'arguing': 1, 'might': 1, 'able': 1, 'bre
ak': 1, 'medieval': 1, 'sexist': 1, 'mold': 1, 'prove': 1, 'worth': 1, 'fighter': 1, 'side':
1, 'missing': 1, 'pure': 1, 'showmanship': 1, 'essential': 1, 'element': 1, 'ever': 1, 'expec
ted': 1, 'climb': 1, 'high': 1, 'ranks': 1, 'nothing': 1, 'differentiates': 1, 'something':
1, 'see': 1, 'given': 1, 'saturday': 1, 'morning': 1, 'subpar': 1, 'instantly': 1, 'forgettab
le': 1, 'songs': 1, 'poorly': 1, 'integrated': 1, 'computerized': 1, 'footage': 1, 'compare':
1, 'run': 1, 'angry': 1, 'ogre': 1, 'herc': 1, 'battle': 1, 'hydra': 1, 'rest': 1, 'case': 1,
'characters': 1, 'stink': 1, 'none': 1, 'remotely': 1, 'interesting': 1, 'film': 1, 'become
s': 1, 'race': 1, 'one': 1, 'bland': 1, 'others': 1, 'end': 1, 'tie': 1, 'win': 1, 'comedy':
1, 'shtick': 1, 'awfully': 1, 'cloying': 1, 'least': 1, 'shows': 1, 'signs': 1, 'pulse': 1,
'fans': 1, "-'": 1, '90s': 1, 'tgif': 1, 'television': 1, 'line': 1, 'thrilled': 1, 'find':
1, 'jaleel': 1, 'urkel': 1, 'white': 1, 'bronson': 1, 'balki': 1, 'pinchot': 1, 'sharing': 1,
'scenes': 1, 'nicely': 1, 'realized': 1, 'though': 1, 'loss': 1, 'recall': 1, 'enough': 1, 's
pecific': 1, 'actors': 1, 'providing': 1, 'voice': 1, 'talent': 1, 'enthusiastic': 1, 'paire
d': 1, 'singers': 1, 'sound': 1, 'thing': 1, 'like': 1, 'big': 1, 'musical': 1, 'moments': 1,
'jane': 1, 'seymour': 1, 'celine': 1, 'dion': 1, 'must': 1, 'strain': 1, 'mess': 1, 'good':
1, 'aside': 1, 'fact': 1, 'children': 1, 'probably': 1, 'bored': 1, 'watching': 1, 'adults':
1, 'grievous': 1, 'error': 1, 'complete': 1, 'lack': 1, 'personality': 1, 'learn': 1, 'goes':
1, 'long': 1, 'way': 1}, 'neg')
```

In [47]:
```python
positive_features = [
    (build_bag_of_words_features_filtered(movie_reviews.words(fileids=[f])), 'pos') \
    for f in positive_fileids
]
```

In [48]:
```python
print(positive_features[6])
```
```
({'apparently': 1, 'director': 1, 'tony': 1, 'kaye': 1, 'major': 1, 'battle': 1, 'new': 1, 'l
ine': 1, 'regarding': 1, 'film': 1, 'american': 1, 'history': 1, 'x': 1, 'know': 1, 'detail
s': 1, 'fight': 1, 'seems': 1, 'happy': 1, 'final': 1, 'product': 1, 'nearly': 1, 'removed':
1, 'name': 1, 'credits': 1, 'altogether': 1, 'heard': 1, 'kind': 1, 'thing': 1, 'happening':
1, 'makes': 1, 'wonder': 1, 'much': 1, 'input': 1, 'studio': 1, 'films': 1, 'produce': 1, 'fo
und': 1, 'extremely': 1, 'good': 1, 'focused': 1, 'look': 1, 'touchy': 1, 'subject': 1, 'raci
sm': 1, 'powerful': 1, 'charismatic': 1, 'performance': 1, 'edward': 1, 'norton': 1, 'hard':
1, 'believe': 1, 'two': 1, 'years': 1, 'since': 1, 'fantastic': 1, 'role': 1, 'primal': 1, 'f
ear': 1, 'starring': 1, 'making': 1, 'star': 1, 'one': 1, 'performers': 1, 'becomes': 1, 'cha
racter': 1, 'work': 1, 'best': 1, 'performances': 1, 'year': 1, 'plays': 1, 'young': 1, 'ma
n': 1, 'named': 1, 'derek': 1, 'vinyard': 1, 'skinhead': 1, 'living': 1, 'venice': 1, 'beac
h': 1, 'brother': 1, 'danny': 1, 'furlong': 1, 'mother': 1, 'beverly': 1, 'angelo': 1, 'siste
r': 1, 'davin': 1, 'jennifer': 1, 'lien': 1, 'opens': 1, 'flashback': 1, 'brutally': 1, 'kill
s': 1, 'black': 1, 'men': 1, 'vandalizing': 1, 'car': 1, 'find': 1, 'lands': 1, 'prison': 1,
'point': 1, 'seen': 1, 'eyes': 1, 'present': 1, 'time': 1, 'high': 1, 'school': 1, 'eager':
1, 'follow': 1, 'footsteps': 1, 'told': 1, 'see': 1, 'path': 1, 'leads': 1, 'adoption': 1, 'w
hite': 1, 'supremacy': 1, 'released': 1, 'served': 1, 'three': 1, 'finds': 1, 'full': 1, 'blo
wn': 1, 'however': 1, 'given': 1, 'violence': 1, 'tries': 1, 'get': 1, 'understand': 1, 'come
s': 1, 'bad': 1, 'things': 1, 'interesting': 1, 'stupid': 1, 'thoughtless': 1, 'people': 1,
'--': 1, 'intelligent': 1, 'articulate': 1, 'voice': 1, 'beliefs': 1, 'disturbingly': 1, 'str
aightforward': 1, 'terms': 1, 'make': 1, 'controversial': 1, 'movie': 1, 'preach': 1, 'righ
t': 1, 'note': 1, 'material': 1, 'mainstream': 1, 'redemption': 1, 'phase': 1, 'main': 1, 'ma
y': 1, 'think': 1, 'way': 1, 'sympathetic': 1, 'partially': 1, 'disagree': 1, 'although': 1,
'advocate': 1, 'presents': 1, 'loud': 1, 'obnoxious': 1, 'also': 1, 'smart': 1, 'reasons': 1,
'believable': 1, 'father': 1, 'arbitrarily': 1, 'killed': 1, 'group': 1, 'clear': 1, 'passion
ate': 1, 'punk': 1, 'looking': 1, 'excuse': 1, 'beat': 1, 'course': 1, 'helps': 1, 'actor':
1, 'talented': 1, 'play': 1, 'part': 1, 'astonishing': 1, 'frightening': 1, 'looks': 1, 'shav
ed': 1, 'head': 1, 'swastika': 1, 'chest': 1, 'addition': 1, 'getting': 1, 'perfect': 1, 'req
uires': 1, 'intelligence': 1, 'depth': 1, 'whole': 1, 'lot': 1, 'shouting': 1, 'ease': 1, 'ev
en': 1, 'meanest': 1, 'likable': 1, 'quality': 1, 'gutsy': 1, 'approach': 1, 'telling': 1, 's
tory': 1, 'adds': 1, 'subplot': 1, 'principal': 1, 'avery': 1, 'brooks': 1, 'obsessed': 1, 'p
urging': 1, 'hatred': 1, 'terrific': 1, 'standouts': 1, 'visually': 1, 'indulges': 1, 'artist
ic': 1, 'choices': 1, 'nicely': 1, 'lots': 1, 'slow': 1, 'motion': 1, 'strange': 1, 'camera':
1, 'angles': 1, 'add': 1, 'moody': 1, 'atmosphere': 1, 'like': 1, 'movies': 1, 'lately': 1,
'skims': 1, 'past': 1, 'greatness': 1, 'last': 1, 'minutes': 1, 'climactic': 1, 'scene': 1,
'moving': 1, 'picture': 1, 'ends': 1, 'pretentious': 1, 'preachy': 1, 'resolution': 1, 'featu
ring': 1, 'brief': 1, 'narration': 1, 'subtle': 1, 'felt': 1, 'slap': 1, 'face': 1, 'hand':
1, 'fed': 1, 'theme': 1, 'simplistic': 1, 'exactly': 1, 'disliked': 1, 'version': 1, 'perhap
s': 1, 'problem': 1, 'imagine': 1, 'least': 1, 'pleased': 1, 'many': 1, 'timid': 1, 'weak':
1, 'manages': 1, 'compelling': 1, 'argument': 1, 'without': 1, 'advocating': 1}, 'pos')
```

In [49]:
```python
from nltk.classify import NaiveBayesClassifier
```

In [50]:
```python
split = 800
```

In [51]:
```python
sentiment_classifier = NaiveBayesClassifier.train(positive_features[:split]+negative_feature
s[:split])
```

In [52]:
```python
nltk.classify.util.accuracy(sentiment_classifier, positive_features[:split]+negative_feature
s[:split])*100
```