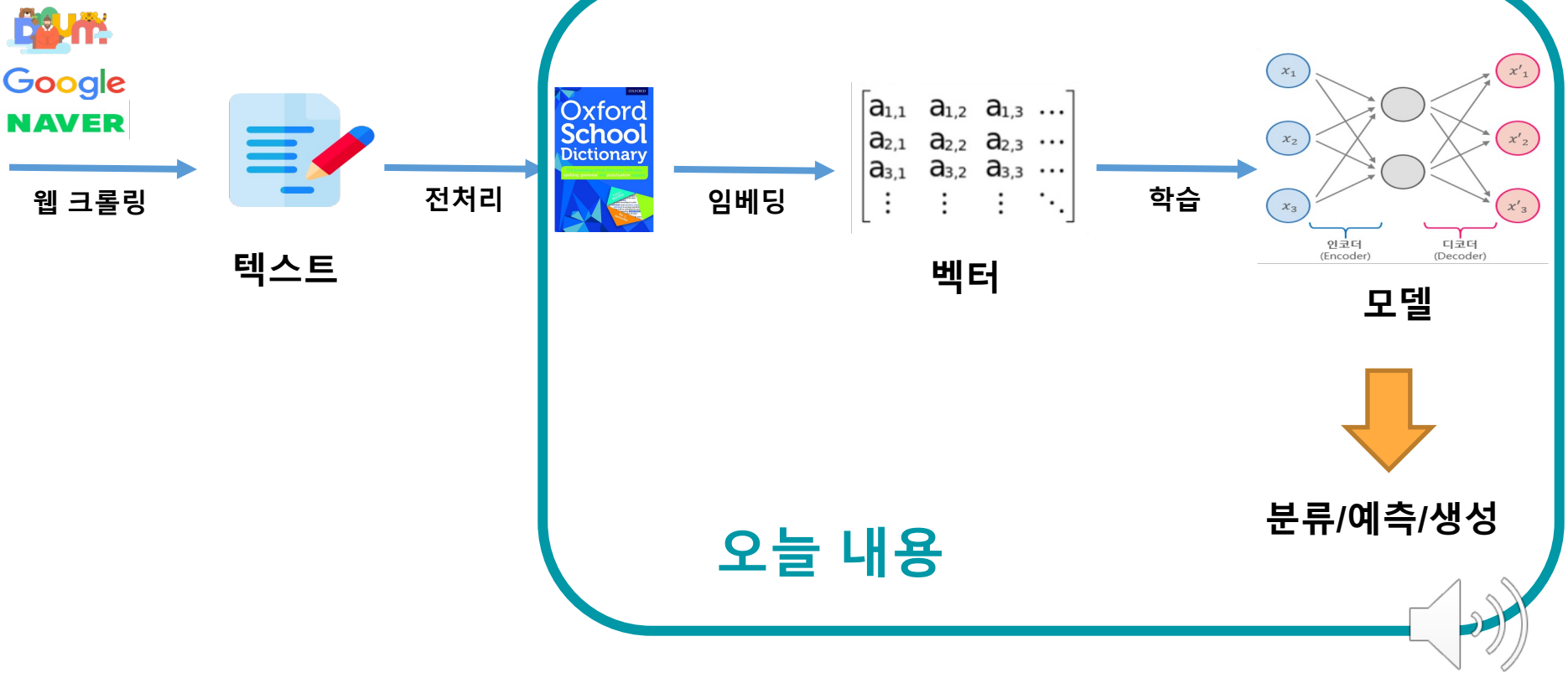
Text Classification

SLP Ch. 4

https://web.stanford.edu/~jurafsky/slp3/4.pdf

자연어 처리 과정


Google
NAVER
웹 크롤링
텍스트
전처리
Oxford School Dictionary
임베딩
학습
인코더 (Encoder)
디코더 (Decoder)
벡터
모델
분류/예측/생성
오늘 내용

Outline

- Text Classification
- Bayesian classification
- Sentiment Analysis

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

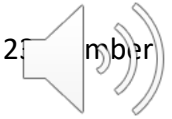
Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

Male or female author?

1. By 1925 present-day Vietnam was divided into three parts under French colonial rule. The southern region embracing Saigon and the Mekong delta was the colony of Cochin-China; the central area with its imperial capital at Hue was the protectorate of Annam…

2. Clara never failed to be astonished by the extraordinary felicity of her own name. She found it hard to trust herself to the mercy of fate, which had managed over the years to convert her greatest shame into one of her greatest assets…

S. Argamon, M. Koppel, J. Fine, A. R. Shimoni, 2003. "Gender, Genre, and Writing Style in Formal Written Texts," Text, volume 23, number 3, pp. 321–346



Positive or negative movie review?

- Unbelievably disappointing
- Full of zany characters and richly applied satire, and some great plot twists
- This is the greatest screwball comedy ever filmed
- It was pathetic. The worst part about it was the boxing scenes.

What is the subject of this article?

?

MeSH Subject Category Hierarchy

Antogonists and Inhibitors
Blood Supply
Chemistry
Drug Therapy
Embryology
Epidemiology
…

Text Classification

- Sentiment analysis
- Genre classification
- Spam detection
- Topic classification
- Authorship identification, age/gender identification
- Subjectivity detection identifies the parts of a text that express subjective opinions as well as other non-factual content such as speculation and hypotheticals
- Emotion classification identifies happiness, surprise, fear, sadness, anger, and contempt

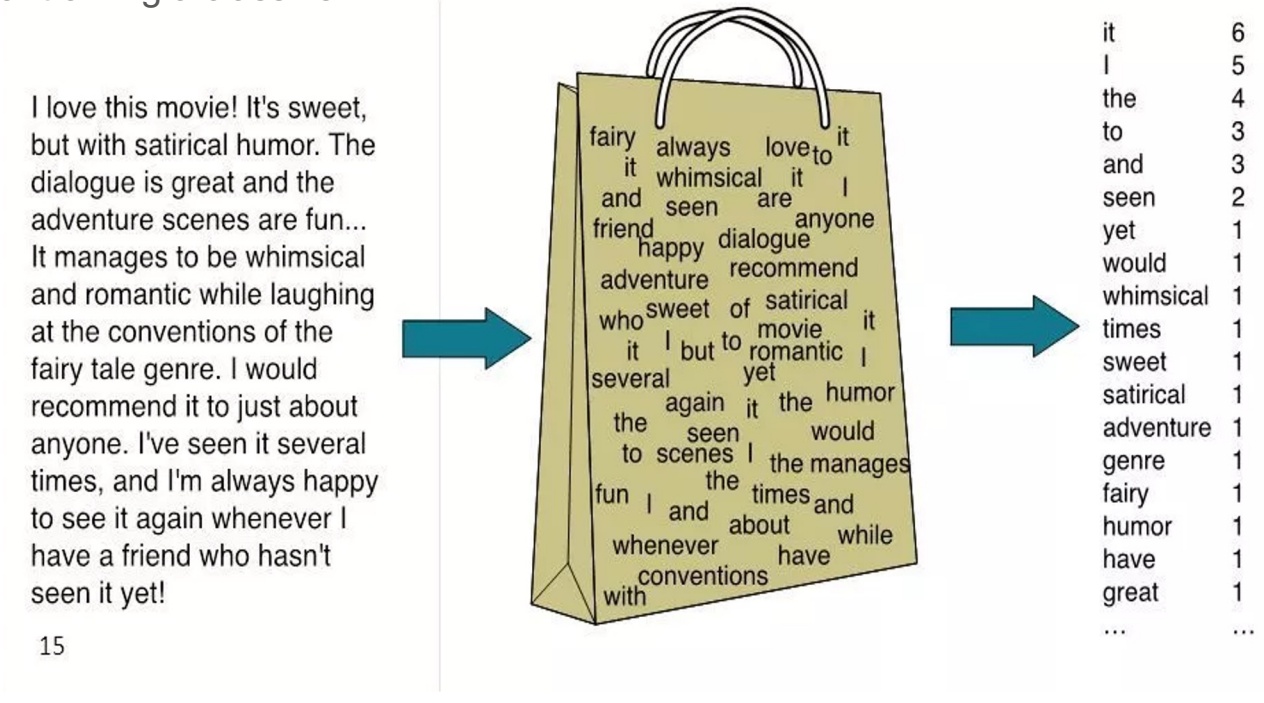
Naïve Bayes Classification

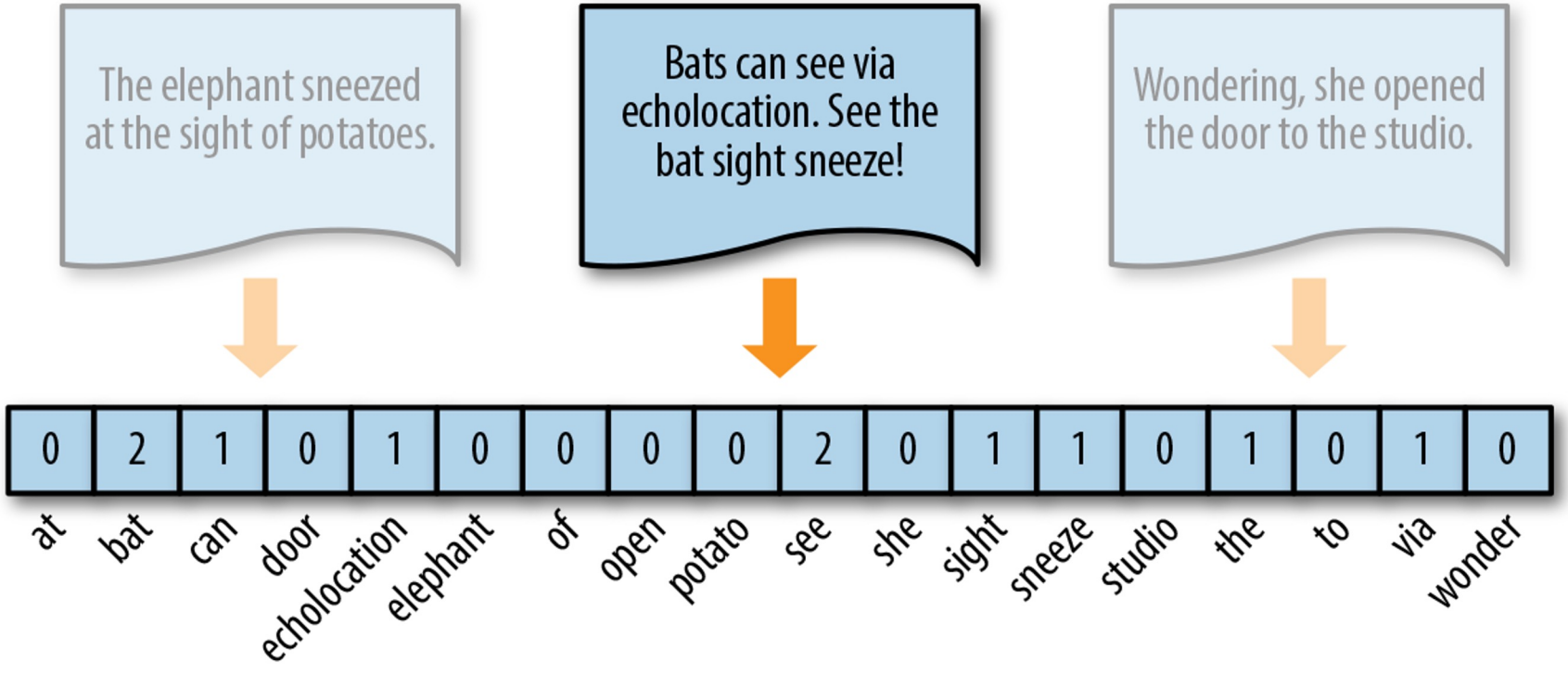
Naïve Bayes Intuition

- Simple (“naïve”) classification method based on Bayes rule
- **Conditional independence assumption** (words are conditionally independent of each other)
- Relies on a simple representation of document: *Bag of words*

BOW (Bag-of-words)

- A sentence or a document is represented as the bag of its words, disregarding grammar and word order but keeping multiplicity
- Used for document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier





출처: Applied Text Analysis with Python - Bengfort, Bilbro, Ojeda (O'REilly)

The bag of words representation

(1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.

(1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
(2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]

BoW1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
BoW2 = {"John":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};

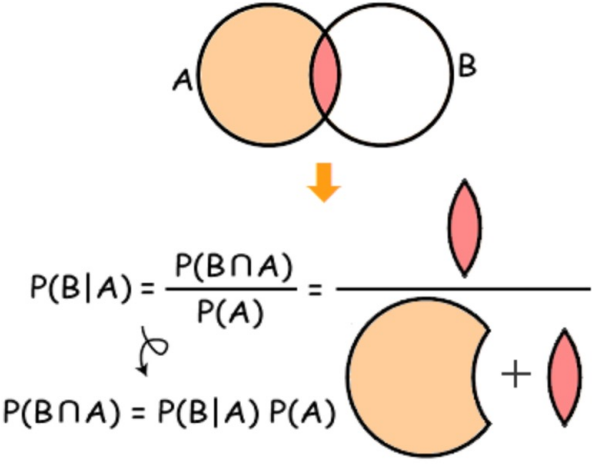
Bayes' Rule

For a document ***d*** and a class ***c***

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

조건부 확률

조건부 확률 P(B | A)는 어떤 사건 A가 일어났을 때 사건 B가 일어날 확률을 의미한다. 이때 사건 A가 발생했을 때 사건 B가 발생할 확률은 사건 A의 영향을 받아 변한다.



확률 P(B | A)는 사건 A가 일어났다는 조건 아래서 사건 B가 일어날 확률으로 **'사건 A가 일어났다는 조건 아래서**'라는 의미는 **표본공간이 A로 축소됨을 뜻한다**.

출처: https://m.blog.naver.com/alwaysneoi/100148922781

Naïve Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is “maximum a posteriori”
= most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule $P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c)P(c)$$

Document d represented as words $x_1..x_n$

Naive Bayesian Classification

단어 x,y로 이루어진 문서가 분류 1에 속할 확률이 p1(x,y)이고, 분류 2에 속할 확률이 p2(x,y)일때,

- p1(x,y) > p2(x,y) 이면, 이 문서는 분류 1에 속함
- p1(x,y) < p2(x,y) 이면, 이 문서는 분류 2에 속함

즉, 문서가 **각 분류에 속할 확률을 측정하여, 확률이 큰 쪽으로 예측**

Sentiment analysis as a document classification

- Labels: POSITIVE, NEGATIVE, (NEUTRAL)
- Labels could be annotated by hand, or obtained automatically
  - Tweets containing happy emoticons marked as positive, sad emoticons as negative
  - Reviews with four or more stars marked as positive, three or fewer stars as negative
- Applications
  - Marketers are interested to know how people respond to advertisements, services, and products
  - Social scientists are interested in how emotions are affected by phenomena such as the weather, and how both opinions and emotions spread over social networks

  - E.g., reviews of movies and restaurants
  - + ...awesome caramel sauce and sweet toasty almonds. I love this place!
  - − ...awful pizza and ridiculously overpriced...

Sentiment classification when probability is given

- Assigning each word: P(word | c)
- Assigning each sentence: P(s|c)=∏ P(word|c)

Class *pos*

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film
…	

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1

P(s | pos) = 0.0000005

Sentiment classification

Which class assigns the higher probability to *s*?

Model pos

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg

0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

$$P(s|\text{pos}) > P(s|\text{neg})$$

Problem: Genre Classification from movie reviews

영화평이 “fast,furious,fun” 라는 문서로 표현될 때,

Comedy일 확률: P(Comedy | Words) =
P(Words | Comedy) X P(Comedy) / P(Words) -> A
Action 확률: P(Action | Words) =
P(Words | Action) X P(Action) / P(Words) -> B

영화	단어	분류
1	fun,couple,love,love	Comedy
2	fast,furious,shoot	Action
3	Couple,fly,fast,fun,fun	Comedy
4	Furious,shoot,shoot,fun	Action
5	Fly,fast,shoot,love	Action

A > B라면 Comedy로 분류하고, A < B라면 Action으로 분류
A, B의대소만 비교 하기 때문에 P(Words) 무시하고 아래만 계산

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

A = P(Words | Comedy) X P(Comedy)
B = P(Words | Action) X P(Action)

http://unlimitedpower.tistory.com/entry/NLP-Naive-Bayesian-Classification나이브-베이즈-분류?category=407624

- P(fast,furious,fun | Comedy)는 Comedy 영화 중, fast, furious, fun이 나타날 확률. 즉, P(fast | Comedy) * P(furious | Comedy) * P(fun | Comedy)
- Comedy 영화에 나오는 총 단어의 개수 : 9
- P(fast | Comedy) * P(furious | Comedy) * P(fun | Comedy) = (1/9) * (0/9) * (3/9)
  5편중에서 2편이 Comedy이므로, P(Comedy) = (2/5)
- P(Comedy | Words) = ((1/9) * (0/9) * (3/9)) * 2/5 = 0
- P(Action | Words) = ((2/11) * (2/11)*(1/11)) * 3/5 = 0.0018 (동일한 방법으로 계산)
- **P(Comedy | Words) <P(Action | Words) 이므로 Action 으로 분류**

영화	단어	분류
1	fun,couple,love,love	Comedy
2	fast,furious,shoot	Action
3	Couple,fly,fast,fun,fun	Comedy
4	Furious,shoot,shoot,fun	Action
5	Fly,fast,shoot,love	Action

각 단어의 빈도 수

Count (fast,comedy) = 1
Count(furious,comedy) = 0
Count(fun,comedy) = 3
Count(fast,action)= 2
Count(furious,action)=2
Count(furious,action) = 1

http://unlimitedpower.tistory.com/entry/NLP-Naive-Bayesian-Classification나이브-베이즈-분류?category=407624

Laplace Smoothing

- 확률이 0이 되는 문제 방지 위해 빈도에 1 추가

$$\hat{P}(x|c) = \frac{count(x,c)+1}{\Sigma_{x \in V}(count(x,c)+1)} = \frac{count(x,c)+1}{(\Sigma_{x \in V} count(x,c)) + |V|}$$

|V|는 전체 단어의 수가 아니라 유일한 단어의 수인 7

- Laplace smoothing을 적용하여 다시 계산

P(Comedy | Words)

$$= \frac{1+1}{9+7} \cdot \frac{0+1}{9+7} \cdot \frac{3+1}{9+7} \cdot \frac{2}{5}$$

$$= \frac{2}{16} \cdot \frac{1}{16} \cdot \frac{4}{16} \cdot \frac{2}{5}$$

$$= 0.00078$$

<P(Action | Words)

$$= \frac{2+1}{11+7} \cdot \frac{2+1}{11+7} \cdot \frac{1+1}{11+7} \cdot \frac{3}{5}$$

$$= \frac{3}{18} \cdot \frac{3}{18} \cdot \frac{2}{18} \cdot \frac{3}{5}$$

$$= 0.0018$$

Log를 이용한 언더 플로우 방지

- 문장의 발생 확률은 각 단어의 확률의 곱으로, 소숫점 아래로 작아져 부동 소수점 문제 발생
- **로그 (log) 적용**

  P(comedy|words) = P(words|comedy)*P(comedy)

  log(P(comedy|words)) = log(P(words|comedy)*P(comedy))

  log(P(words|comedy)*P(comedy)) = log(P(fun|comedy)*P(furios|comedy)*…*P(Comedy))
  =
  log(P(fun|comedy))+log(P(furios|comedy)+…+log(P(Comedy))

http://bcho.tistory.com/1010

Classification Problem: Spam Filtering

- Decides if a particular piece of email is an example of spam
- One of the first applications of Naive Bayes to text classification (Sahami et al., 1998)
- SpamAssassin Tool Features:
  - Mentions “Generic Viagra”, “online pharmaceutical”, “WITHOUT ANY COST”, “Dear Winner”, “One hundred percent guaranteed”
  - Mentions millions of (dollar) ((dollar) NN,NNN,NNN.NN)
  - From: starts with many numbers
  - Subject is all capitals
  - HTML has a low ratio of text to image area
  - Claims you can be removed from the list

https://spamassassin.apache.org/

Lexicon-based classification Limitations

- Classifies documents by counting words against positive and negative sentiment words
- Can make bag-of-words classification ineffective: ***negation*** and ***irrealis*** — events that are hypothetical or otherwise non-factual
- Solutions
  - That's ***not bad*** for the first day.
  - This is ***not the worst*** thing that can happen.
  - ***It would be nice*** if you acted like you understood.
  - There is ***no reason at all to believe*** that the polluters are suddenly going to become reasonable.
  - This film ***should be*** brilliant.

a bag-of-**bigrams/trigram** model
(*that's*, *not*), (*not*, *bad*), (*bad*, *for*), . . .

AllenNLP: sentiment analysis

- AllenNLP is a NLP platform
- 데모: https://demo.allennlp.org/sentiment-analysis/glove-sentiment-analysis
- Uses GloVe embeddings and is trained on the binary classification setting of the Stanford Sentiment Treebank. It achieves about 87% on the test set.
- Input sentence: ‘I'm not sure if this is good.’

  => The model is somewhat confident that the sentence has a negative sentiment.

Naive Bayes is Not So Naive

- Very fast, low storage requirements
- Very good in domains with many equally important features
- Optimal if the independence assumptions hold
- A good dependable **baseline for text classification**

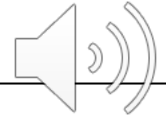
Text Classification using NLTK and Scikit-learn

Ch. 6. Learning to Classify Text
http://www.nltk.org/book/ch06.html

Gender Identification

- Names ending in *a, e* and *i* are likely to be female, while names ending in *k, o, r, s* and *t* are likely to be male
- Decide what **features** of the input are relevant, and **how to encode** those features

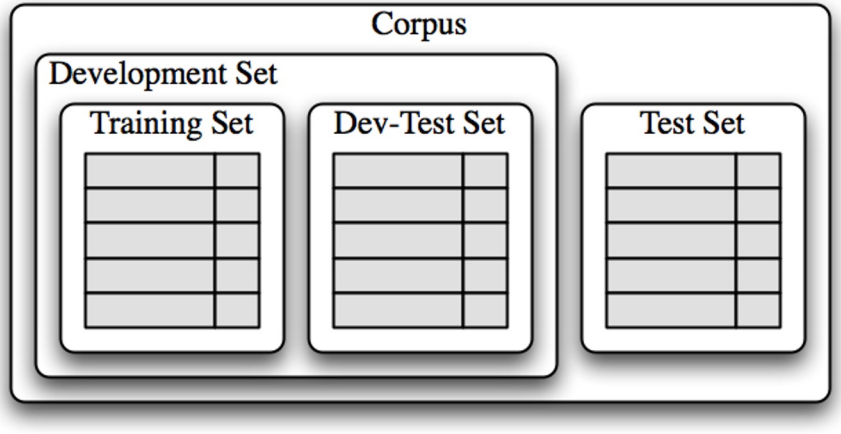
```
def gender_features(word): #builds a dictionary containing last character
    return {'last_letter': word[-1]}
>>> gender_features('Shrek')
{'last_letter': 'k'}

from nltk.corpus import names
import random
labeled_names = ([(name, 'male') for name in names.words('male.txt')] +
... [(name, 'female') for name in names.words('female.txt')])
random.shuffle(labeled_names)
featuresets = [(gender_features(n), gender) for (n, gender) in
labeled_names]
train_set, test_set = featuresets[500:], featuresets[:500]
classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> print(nltk.classify.accuracy(classifier, test_set))
0.77

>>> classifier.show_most_informative_features(5)
        Most Informative Features
           last_letter = 'a'           female : male   =     33.2 : 1.0
           last_letter = 'k'             male : female =     32.6 : 1.0
           last_letter = 'p'             male : female =     19.7 : 1.0
           last_letter = 'v'             male : female =     18.6 : 1.0
           last_letter = 'f'             male : female =     17.3 : 1.0
```



Choosing The Right Features

- Selecting relevant features and deciding how to encode them can have a big impact on the learning method's ability
    - If you provide too many features, the algorithm will rely on idiosyncrasies of your training data (**overfitting**) and can be problematic when working with small training sets
- To refine the feature set, we select a development set, being subdivided into the training set and the dev-test set
- The training set is used to train the model, and the dev-test set is used to perform error analysis

```
>>> train_names = labeled_names[1500:]
>>> devtest_names = labeled_names[500:1500]
>>> test_names = labeled_names[:500]
```



- Using the dev-test set, we can generate a list of the errors that the classifier makes when predicting name genders
- Looking through this list of errors makes it clear that some suffixes that are more than one letter can be indicative of name genders
    - names ending in *yn* are mostly female, despite the fact that names ending in *n* tend to be male; and names ending in *ch* are usually male, even though names that end in *h* tend to be female

```
>>> train_set = [(gender_features(n), gender) for (n, gender) in train_names]
>>> devtest_set = [(gender_features(n), gender) for (n, gender) in
devtest_names]
>>> test_set = [(gender_features(n), gender) for (n, gender) in test_names]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> print(nltk.classify.accuracy(classifier, devtest_set))
0.75
>>> errors = []
>>> for (name, tag) in devtest_names:
...     guess = classifier.classify(gender_features(name))
...     if guess != tag:
...         errors.append( (tag, guess, name) )
>>> for (tag, guess, name) in sorted(errors):
...     print('correct={:<8} guess={:<8s} name={:<30}'.format(tag, guess, name))
correct=female   guess=male     name=Abigail
correct=female   guess=male     name=Cindelyn
correct=female   guess=male     name=Katheryn
correct=female   guess=male     name=Kathryn
correct=male     guess=female   name=Aldrich
correct=male     guess=female   name=Mitch
correct=male     guess=female   name=Rich
>>> def gender_features(word):
...     return {'suffix1': word[-1:],
...             'suffix2': word[-2:]}
>>> train_set = [(gender_features(n), gender) for (n, gender) in train_names]
>>> devtest_set = [(gender_features(n), gender) for (n, gender) in
devtest_names]
>>> test_set = [(gender_features(n), gender) for (n, gender) in test_names]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> print(nltk.classify.accuracy(classifier, devtest_set))
0.782
```

Document Classification

- Documents labeled with categories
- Movie Reviews Corpus categorizes each review as **positive** or **negative**
- A feature extractor checks whether each of these words is present in a given document
- To check how reliable the resulting classifier is, we compute its accuracy on the test set and use show_most_informative_features() to find out which features the classifier found to be most informative

```
>>> from nltk.corpus import movie_reviews
>>> documents = [(list(movie_reviews.words(fileid)), category)
...              for category in movie_reviews.categories()
...              for fileid in movie_reviews.fileids(category)]
>>> random.shuffle(documents)
all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features

>>> print(document_features(movie_reviews.words('pos/cv957_8737.txt')))
{'contains(waste)': False, 'contains(lot)': False, ...}

featuresets = [(document_features(d), c) for (d,c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
classifier = nltk.NaiveBayesClassifier.train(train_set)

>>> print(nltk.classify.accuracy(classifier, test_set))
0.81
>>> classifier.show_most_informative_features(5)
Most Informative Features
  contains(outstanding) = True           pos : neg    =     11.1 : 1.0
       contains(seagal) = True           neg : pos    =      7.7 : 1.0
  contains(wonderfully) = True           pos : neg    =      6.8 : 1.0
        contains(damon) = True           pos : neg    =      5.9 : 1.0
       contains(wasted) = True           neg : pos    =      5.8 : 1.0
```

Summary

- Many language processing tasks can be viewed as tasks of classification, such as sentiment analysis, spam detection, language identification, and authorship attribution
- Sentiment analysis classifies a text as reflecting the positive or negative orientation that a writer expresses toward some object or a person
- Naive Bayes makes the bag of words assumption (*position doesn't matter*) and the conditional independence assumption (*words are conditionally independent of each other given the class*)
- Naive Bayes with binarized features seems to work better for many text classification tasks