Lecture 6

Text Feature Extraction – Application: SPAM Filtering

Content

- Text Feature Extraction
  - Count Vectorizing
  - N-grams model
  - Co-occurrence matrix
  - TF-IDF
- Application: Spam Filtering
  - Using Naïve bayes for Spam Filtering

1. Feature Extraction

Features

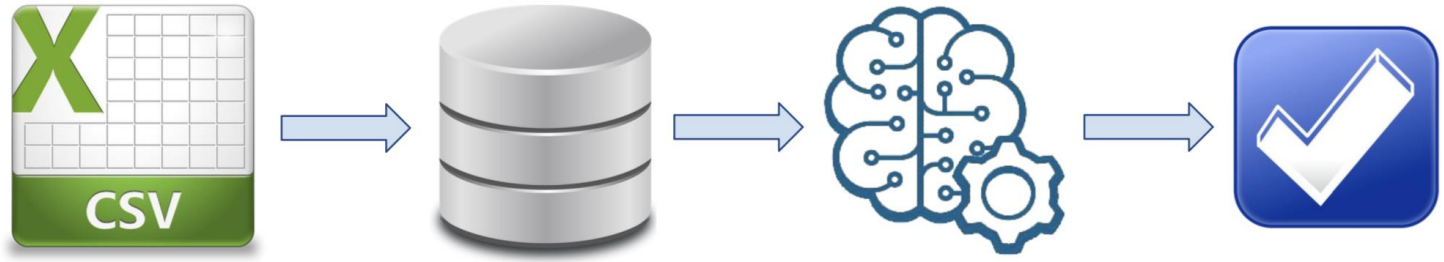
- Given a dataset D of n observations and m attributes
- An attribute is thought of as a set of values describing some aspect across all observations, it is called a ***variable*** - ***feature***
- Each variable represent a measurable characteristic of observations

HR Information		Contact	
Position	**Salary**	**Office**	**Extn.**
Accountant	$162,700	Tokyo	5407
Chief Executive Officer (CEO)	$1,200,000	London	5797
Junior Technical Author	$86,000	San Francisco	1562
Software Engineer	$132,000	London	2558

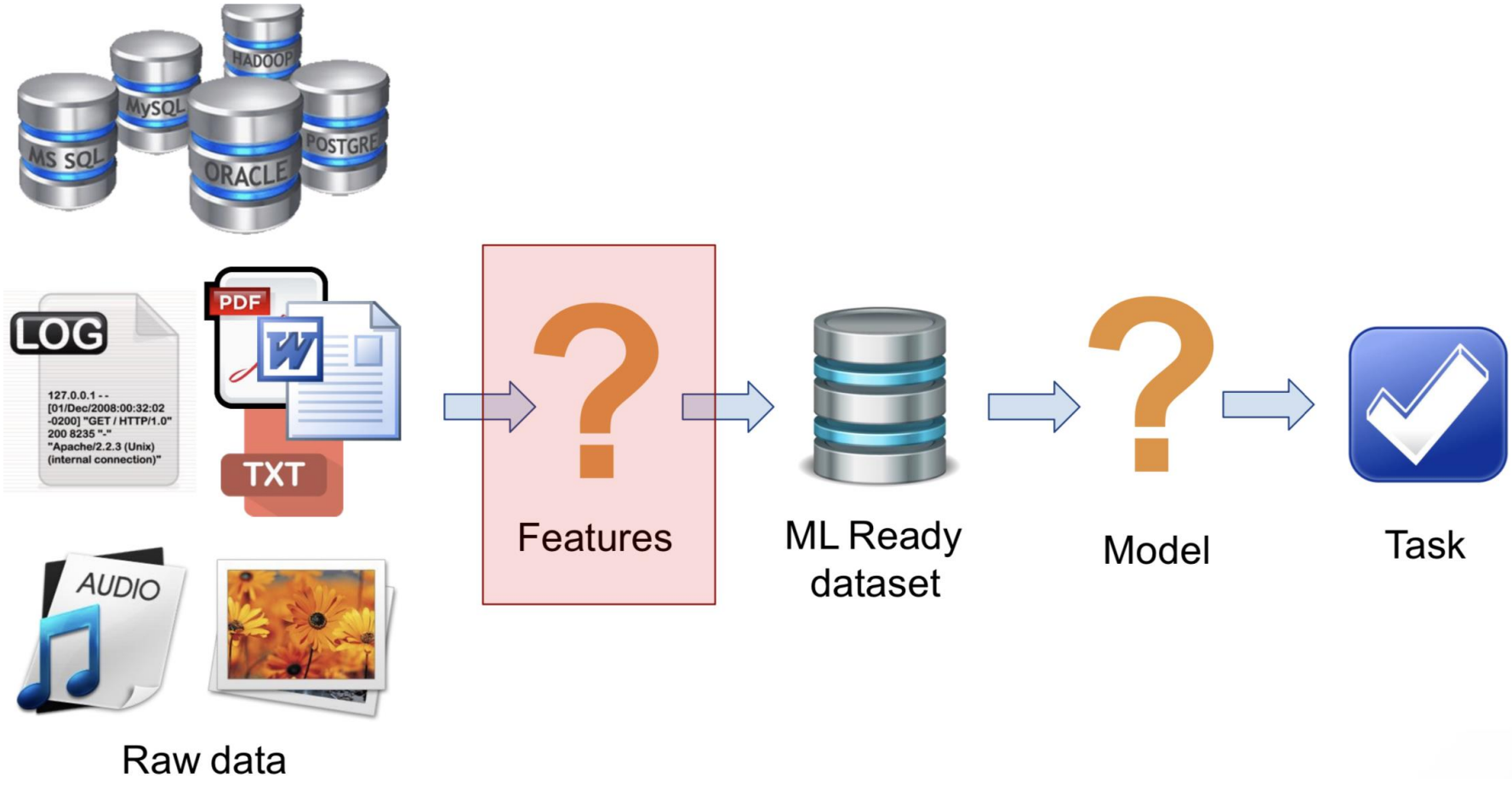
Feature Extraction

"…A process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data…" – Jason Brownlee

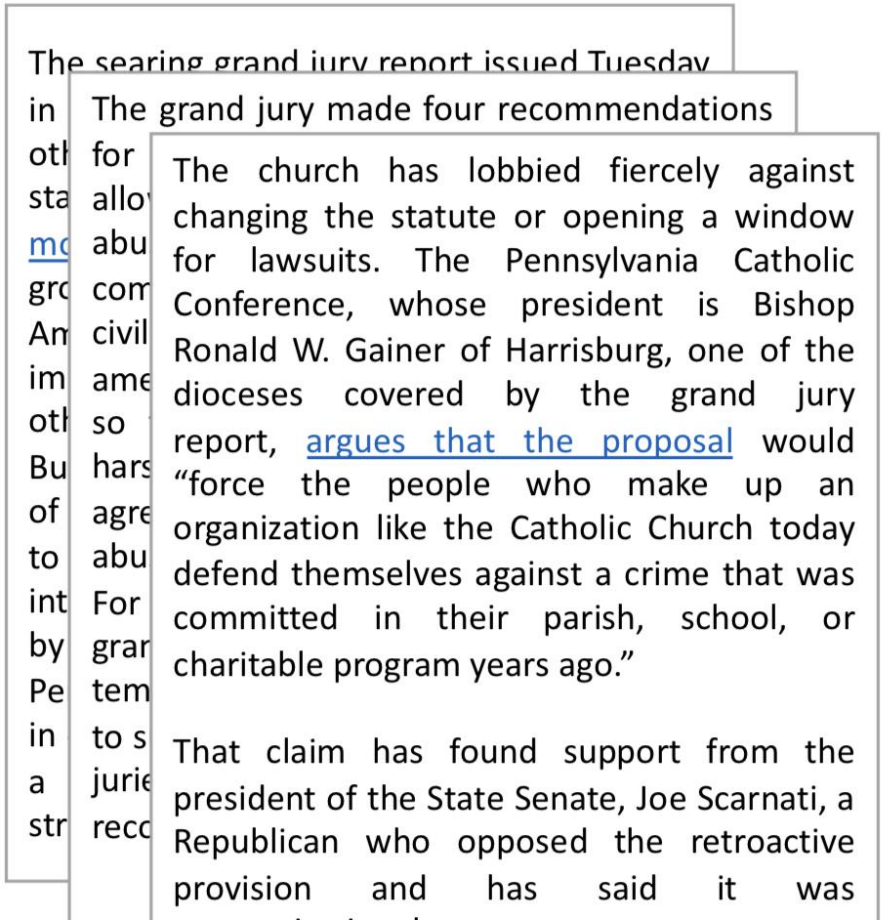
- Ideal:



…but the reality comes with



From text data to features

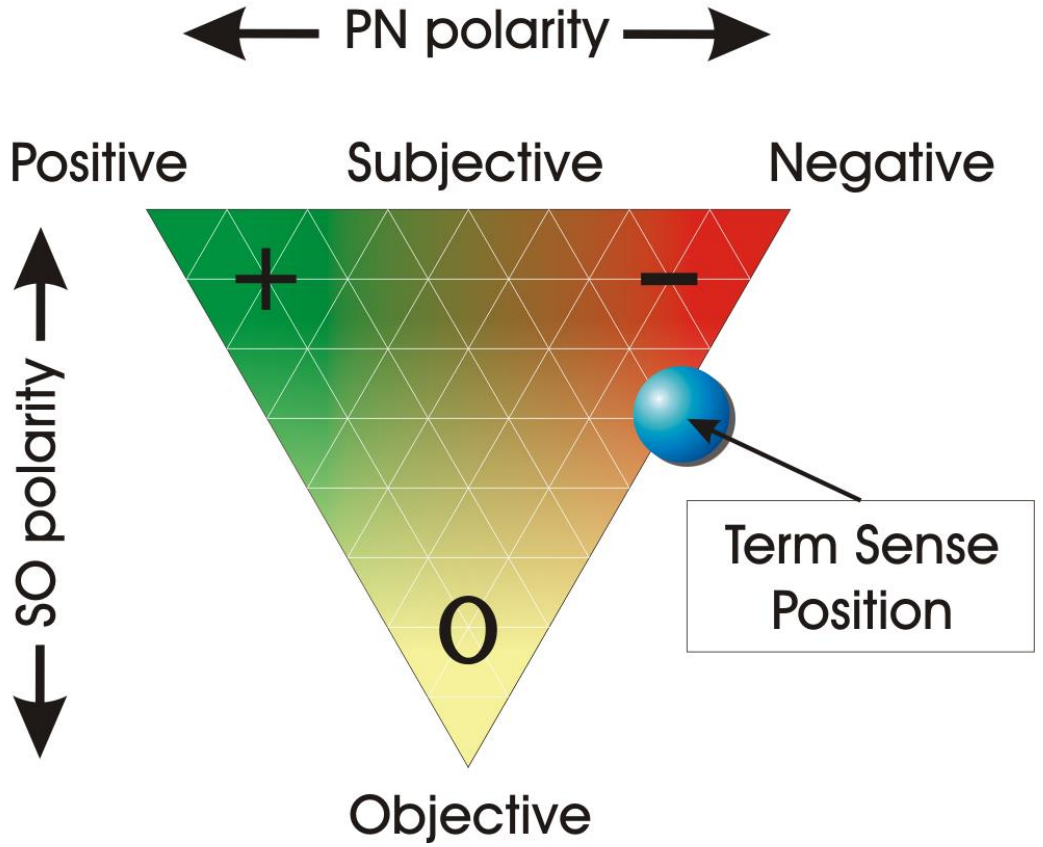


Documents → Vector space representation

	D1	D2	D3	D4	D5
Church	2		3	2	3
Recommendation	3			2	
People		2	3		1
Republican		1		4	

Text Document Representation

- Using statistical features: position, length, etc.
- Using scores from dictionaries:
  - Sentiment dictionaries: SentiWordNet, SentiWords
    - SentiwordNet is a lexicon resource in which each word synset is assigned three scores for sentiment analysis
    - SentiWord using SentiWordNet to assign sentiment score for a word
  - Subjectivity/Objectivity dictionaries: MPQA
- Syntax features: POS tags



Feature Vector (1)

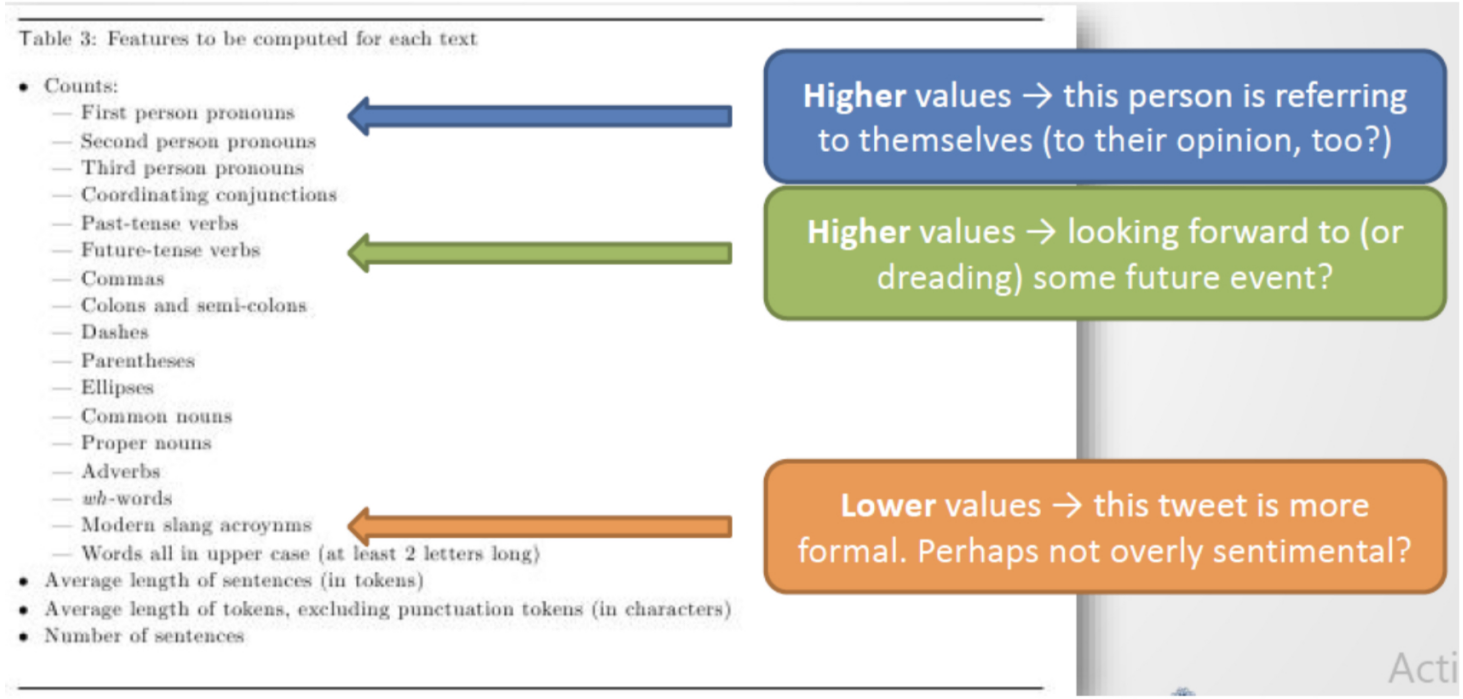
- A vector of values representing features is called feature vector
- For example:

# proper nouns	# 1st person pronouns	# commas
2	0	0
5	0	0
0	1	1

Feature Vector (2)

- Features should allow to distinguish between class labels

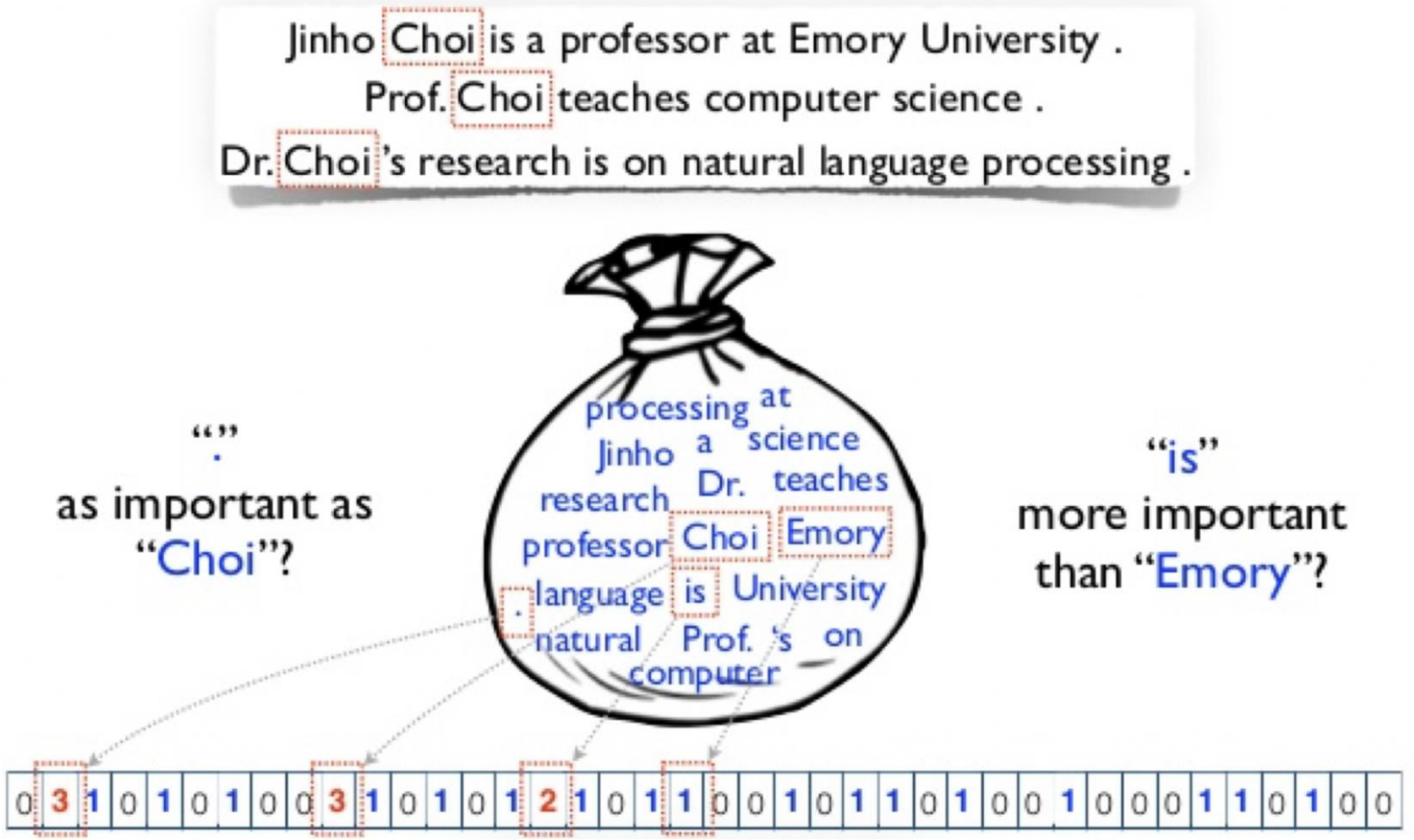
➔ It's important to choose relevant features!



Text document vectorization

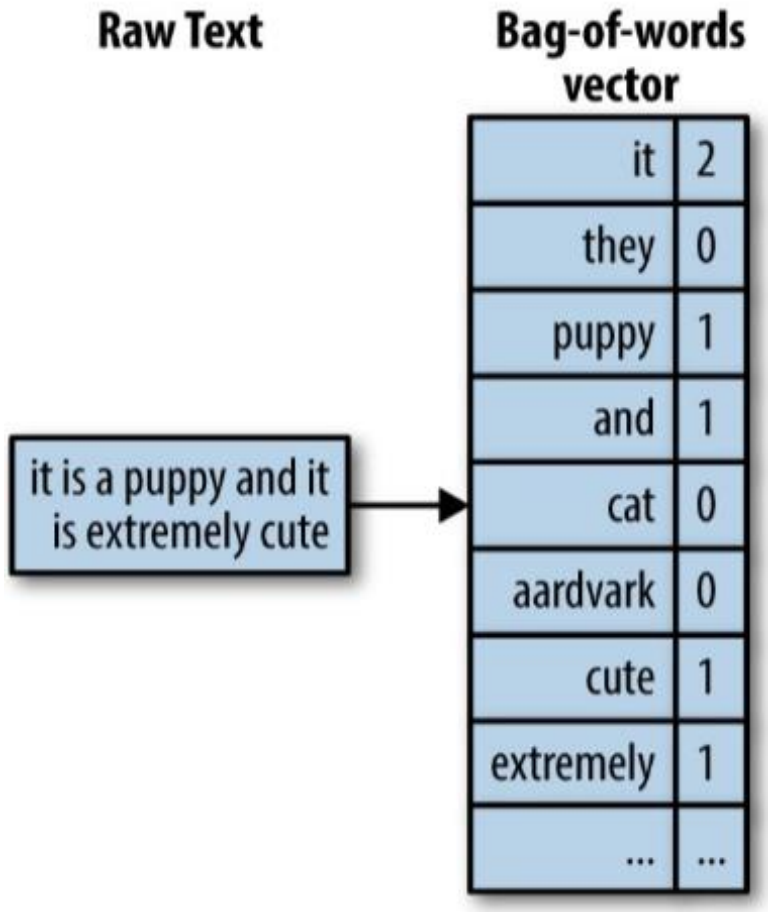
- A process that transform a text document into a numerical vector is called vectorization
- Model: tokenization – counting – normalization
  - **Bag of words**
- Solution: Using `CountVectorizing`

Bag of words


Jinho Choi is a professor at Emory University .
Prof. Choi teaches computer science .
Dr. Choi's research is on natural language processing .
"."
as important as
"Choi"?
processing at
Jinho a science
research Dr. teaches
professor Choi Emory
. language is University
natural Prof. 's on
computer
"is"
more important
than "Emory"?
0 3 1 0 1 0 1 0 0 3 1 0 1 0 1 2 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 0 1 0 0

Bag of words (2)

- Output of the BOW model:
  - A text document is converted into a “flat” vector of counts
  - Does not preserve any original structure
  - The order of words is not taken into account
    - “John is quicker than Mary” and “Mary is quicker than John” have the same vectors!



Bag of n-grams

- A natural extension of BOW model
  - In the BOW model: a word is essentially an unigram
- Bag-of-n-grams representation can be more representative
  - Allow to capture the original structure of the document (through n-grams)
- However, bag-of-n-grams results in a much bigger and sparser feature space

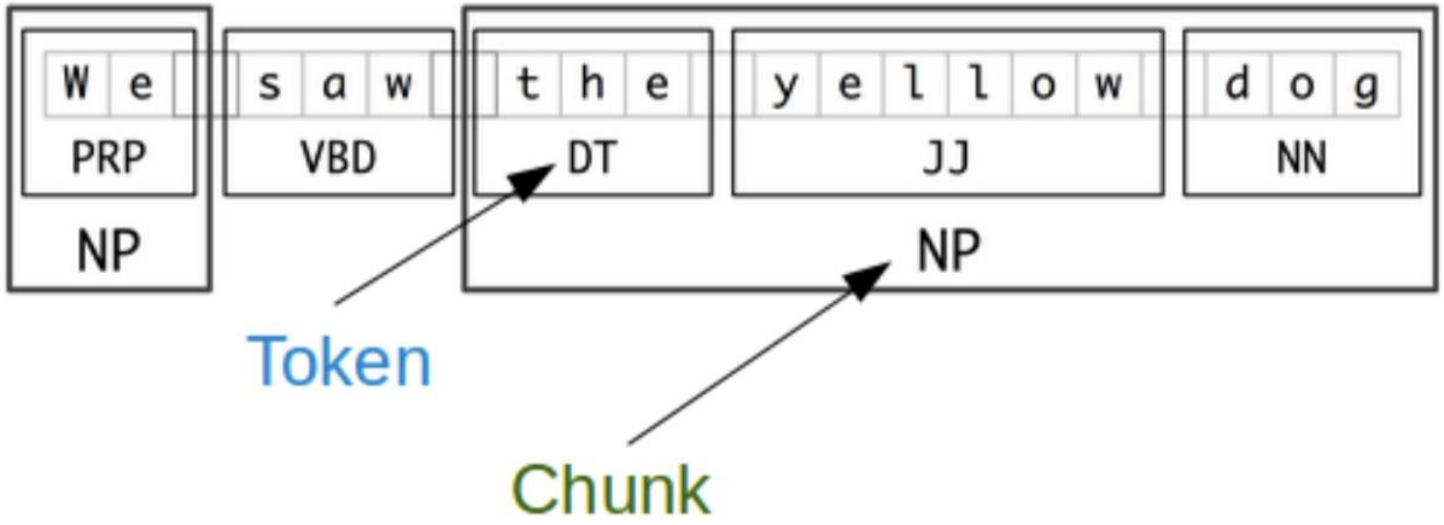
- "I am learning NLP"
- Uni-grams:
  - "I", "am", "learning", "NLP"
- Bi-grams:
  - "I am", "am learning", "learning NLP"

```
text = ["I love NLP and I will learn NLP in 2month "]

{'love nlp': 3, 'nlp and': 4, 'and will': 0, 'will learn': 6,
'learn nlp': 2, 'nlp in': 5, 'in 2month': 1}
[[1 1 1 1 1 1 1]]
```

From Words to N-grams to Phrases

- **Tokenization** to split a string, text into a list of tokens
- **Chunking** a sentences refers to breaking/dividing a sentence into parts of words such as word groups and verb groups



Co-occurrence Matrix

- Counting the co-occurrence of words/sentences or any linguistic elements of text document
- Each row and column of a matrix represents a unique element of a text corpus
- Cells of the matrix represent the number of times two elements appearing together in a pre-defined context

> Apples are green and red.
>
> Red apples are sweet.
>
> Green oranges are sour.

-	apples	are	green	and	red	sweet	oranges	sour
apples	2	2	1	1	2	1	0	0
are	2	3	1	1	2	1	1	1
green	1	1	2	1	1	0	1	1
and	1	1	1	1	1	0	0	0
red	2	2	1	1	2	1	0	0
sweet	1	1	0	0	1	1	0	0
oranges	0	1	1	0	0	0	1	1
sour	0	1	1	0	0	0	1	1

Document Frequency

- Rare terms are more informative than frequent terms
  - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*

→ We want a high weight for rare terms like *arachnocentric*.

TF-IDF (1)

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- df_t is the document frequency of *t*: the number of documents that contain *t*
  - df_t is an inverse measure of the informativeness of *t*
  - $df_t \leq N$
- We define the idf (inverse document frequency) of *t* by

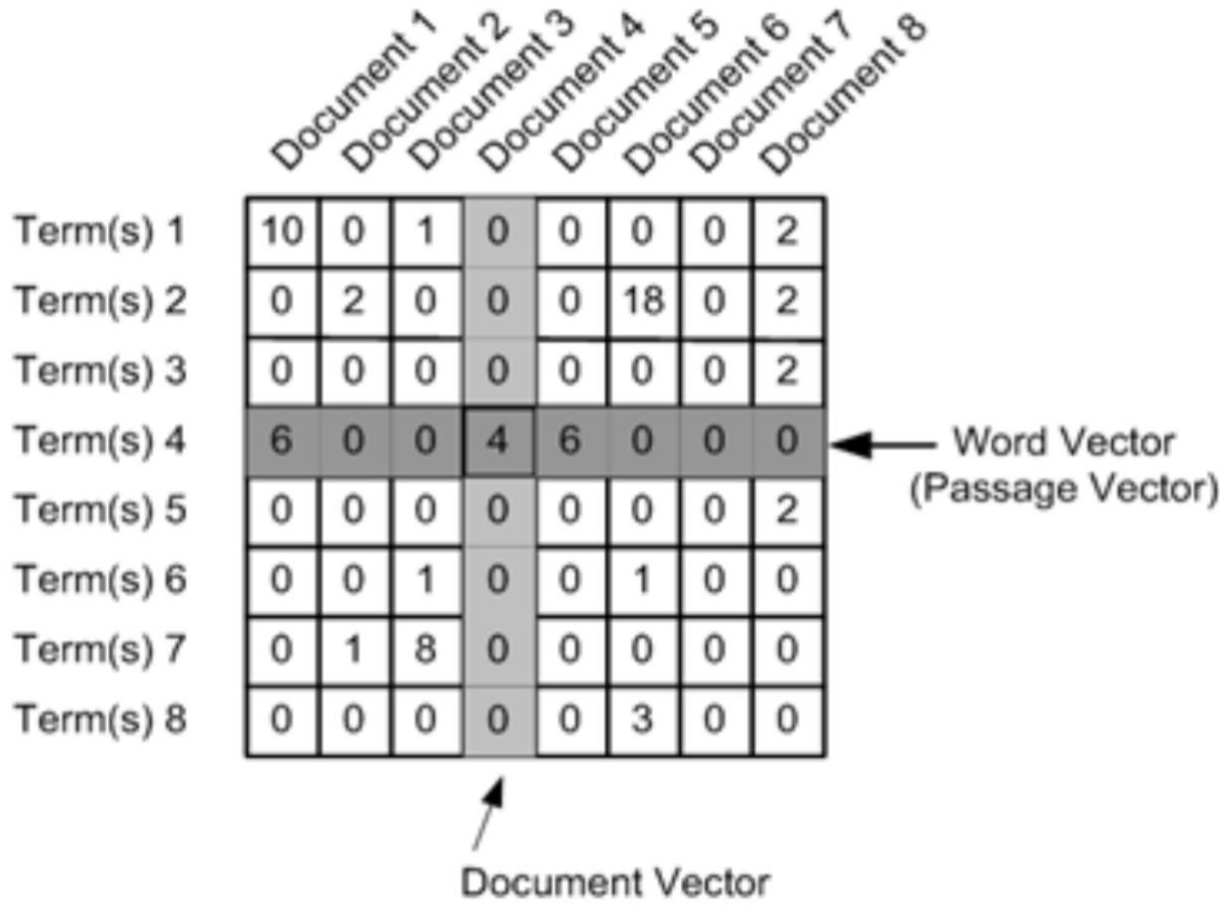
$$idf_t = \log_{10}(N/df_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

TF-IDF(2)

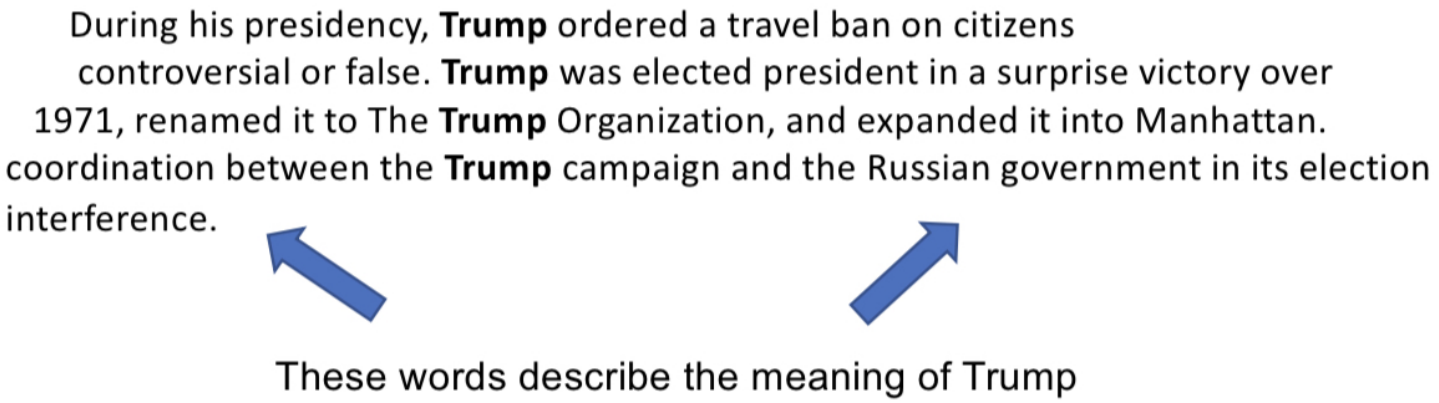
$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents



Word Representation: embedding the context

- Attempt to encode the similarity inside the word vector
- Build on top of the following great idea:
  - “You shall know the word by the company it keeps” – J.R.Firth 1957

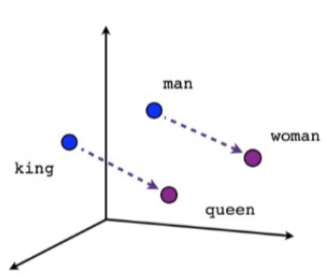


Word Embedding

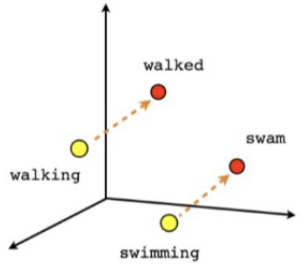
- Each word is encoded in a dense vector (Low dimension)
- Able to capture the semantics
  - Similar words ~ Similar vectors

University =

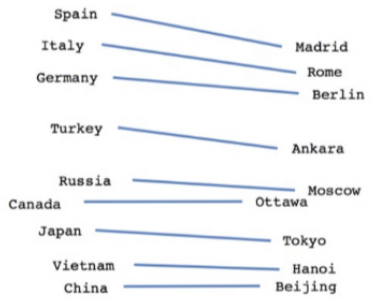
$$\begin{bmatrix} 0.13 \\ 0.67 \\ - \\ 0.34 \\ 0.76 \\ - \\ 0.21 \\ -0.11 \\ - \\ 0.45 \\ 0.87 \\ 0.44 \end{bmatrix}$$



Male-Female



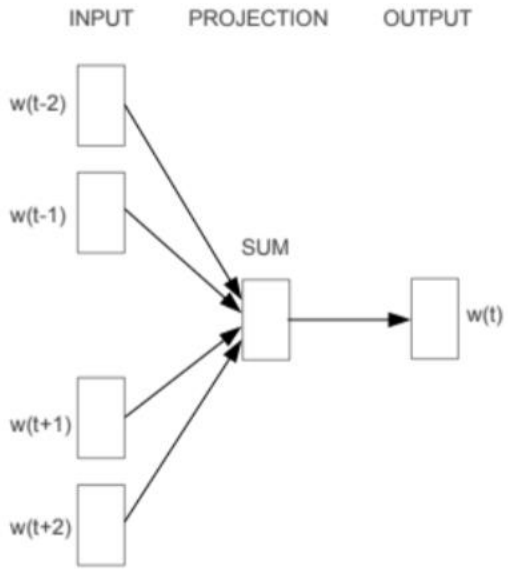
Verb tense



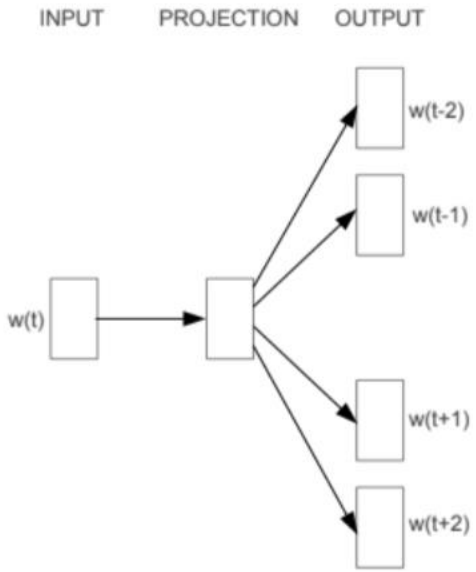
Country-Capital

How to learn word embedding?

- The famous approach: Word2Vec (Mikolov et al. 2013)
- Un-supervised learning
- Large scale dataset
- Lower computation cost
- **High quality word vectors**



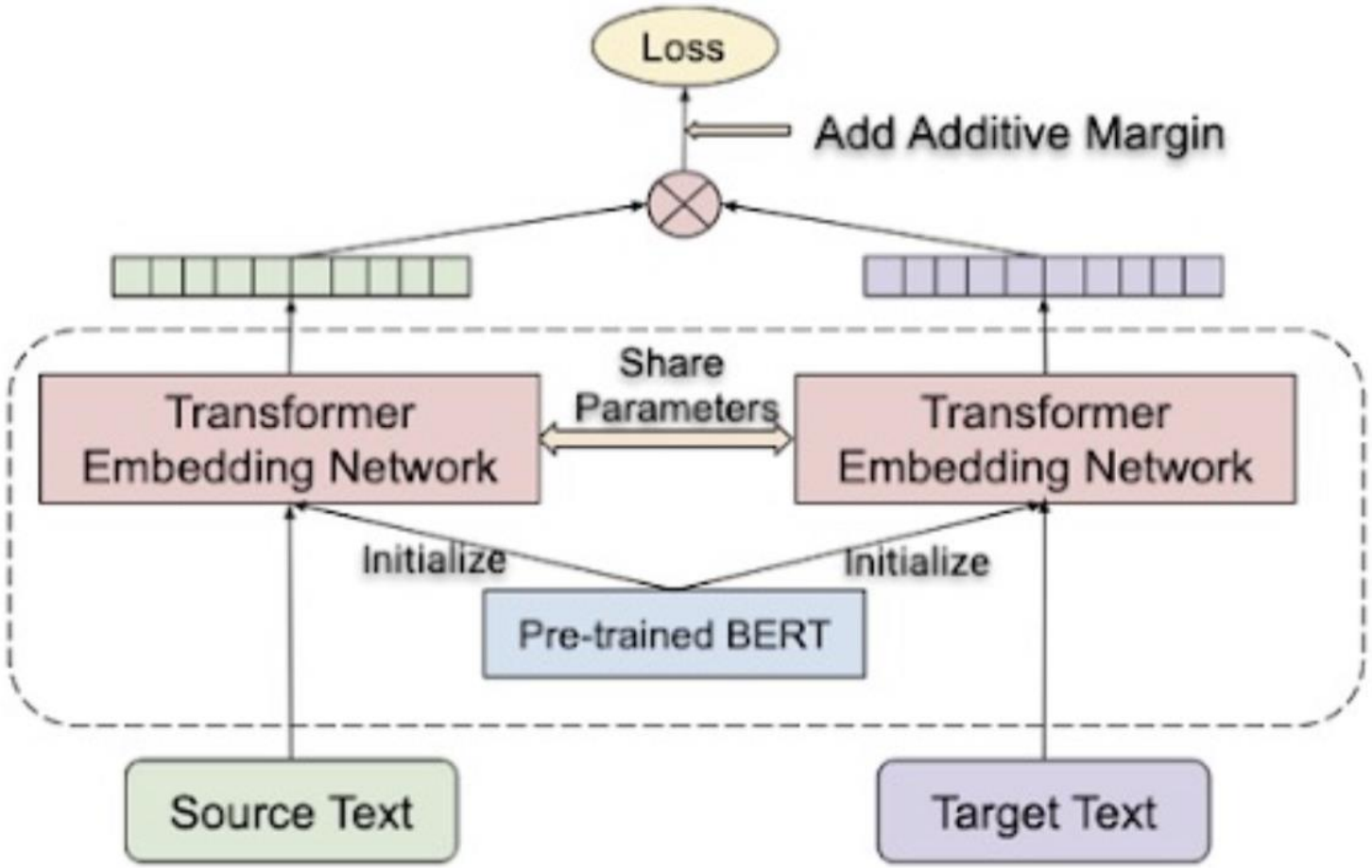
CBOW



Skip-gram

© Mikolov et. Al. 2013

BERT Sentence Embedding

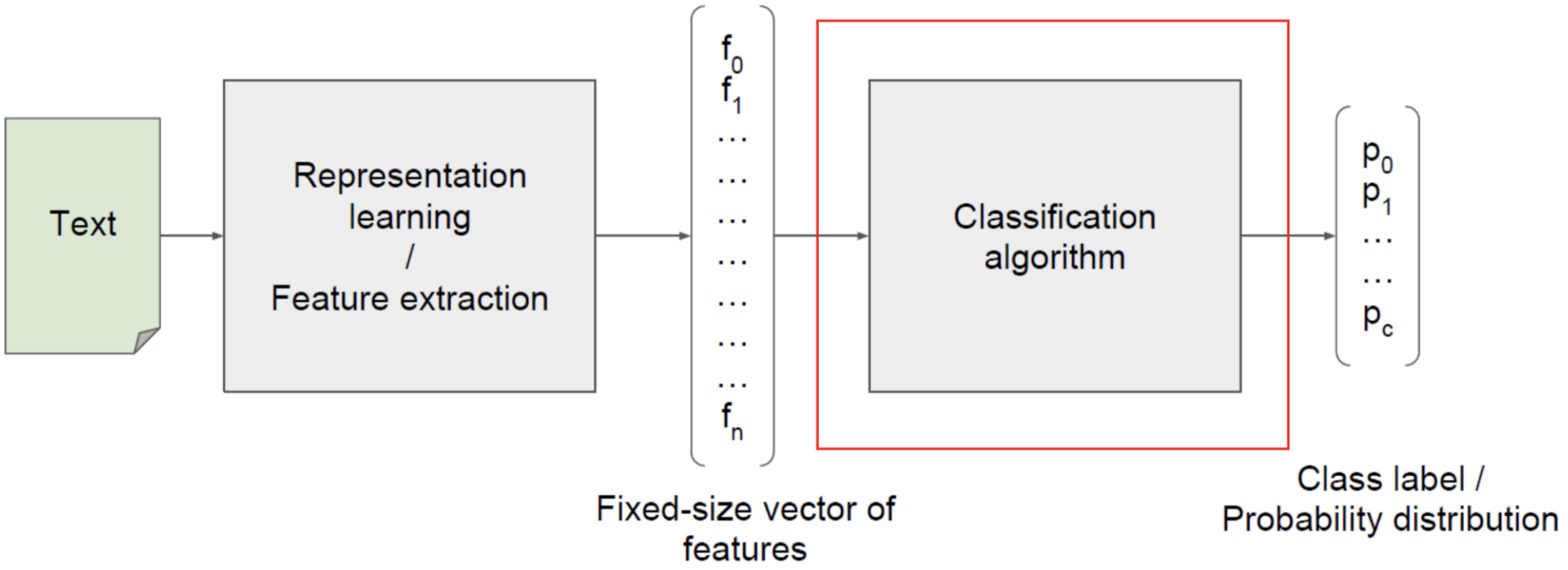


- Feng, Fangxiaoyu, et al. "Language-agnostic BERT Sentence Embedding." *arXiv preprint arXiv:2007.01852* (2020).

2. Text Document Classification

Application: Filtering spam messages

Classification of text documents



Naïve Bayes Method (1)

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Formulate a probabilistic model that places a distribution $P(x, y)$
- Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

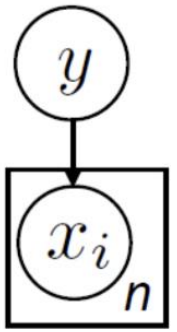
Bayes' Rule

constant: irrelevant for finding the max

$$\propto P(y)P(x|y)$$

"Naive" assumption:

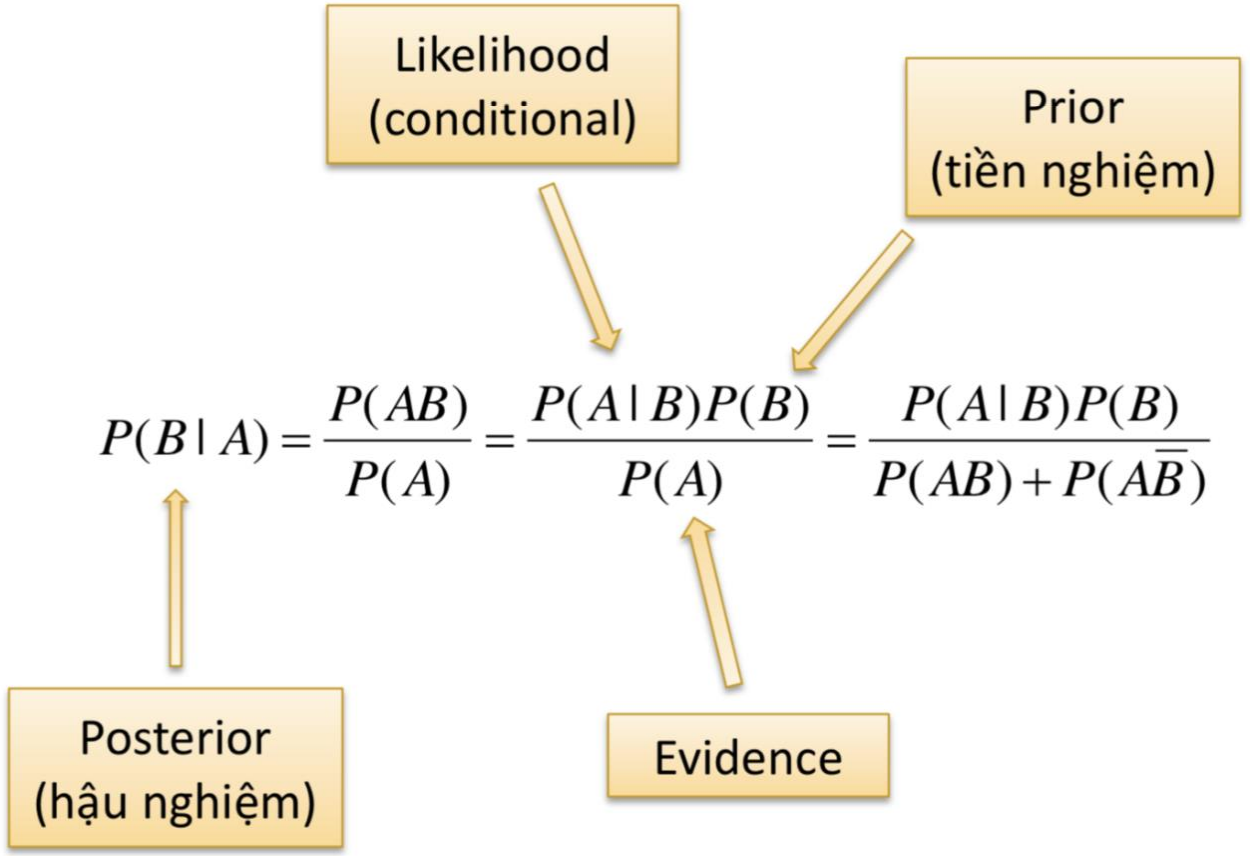
$$= P(y)\prod_{i=1}^{n} P(x_i|y)$$



linear model!

$$\text{argmax}_y P(y|x) = \text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y)\right]$$

Naïve Bayes Method (2)



Posterior = Likelihood * Prior / Evidence

Maximize Likelihood

- Data points (x_j, y_j) provided (*j* indexes over examples)
- Find values of $P(y),\ P(x_i|y)$ that maximize data likelihood (generative):

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j) \right]$$

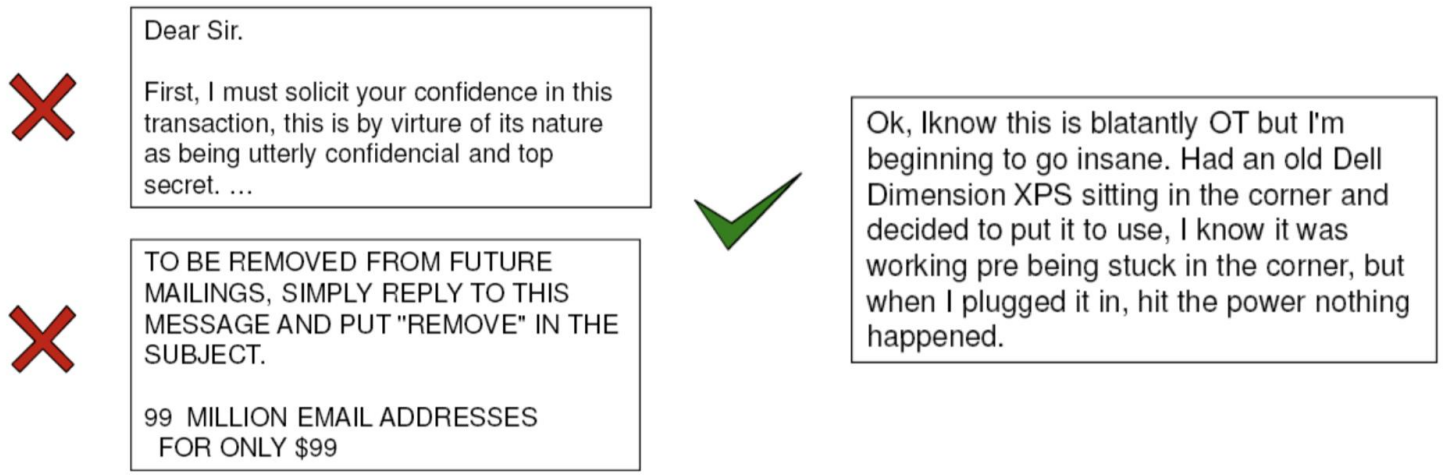
data points (*j*) features (*i*) *i*th feature of *j*th example

- Equivalent to maximizing logarithm of data likelihood:

$$\sum_{j=1}^{m} \log P(y_j, x_j) = \sum_{j=1}^{m} \left[\log P(y_j) + \sum_{i=1}^{n} \log P(x_{ji}|y_j) \right]$$

Naïve Bayes Method for Document Classification (1)

- Objective: assign a document to a class which has the maximum posterior probability – P(class|document)
- For example: classify spam mails
  - An email is spam if P(spam|message) > P(not-spam|message)



Naïve Bayes Method for Document Classification (2)

- Objective: assign a document to a class which has the maximum posterior probability – P(class|document)
- As

$$P(class \mid document) \propto P(document \mid class)P(class)$$

- The model needs to estimate the posterior probability and prior probability
- Estimate the likelihood:
  - Supposing the bag-of-words model allows to represent a document as a set of words

$$P(document \mid class) = P(w_1, \ldots, w_n \mid class) = \prod_{i=1}^{n} P(w_i \mid class)$$

Parameter Estimation (1)

- The parameters of the model include likelihoods P(word|class) and the prior probability P(class)
- How to estimate these parameters for the model?

prior

spam:	0.33
¬spam:	0.67

P(word | spam)

the :	0.0156
to :	0.0153
and :	0.0115
of :	0.0095
you :	0.0093
a :	0.0086
with:	0.0080
from:	0.0075
...	

P(word | ¬spam)

the :	0.0210
to :	0.0133
of :	0.0119
2002:	0.0110
with:	0.0108
from:	0.0107
and :	0.0105
a :	0.0100
...	

Parameter Estimation (2)

- The parameters of the model include likelihoods P(word|class) and the prior probability P(class)
- Training data

$$P(\text{word} \mid \text{class}) = \frac{\#\text{ of occurrences of this word in docs from this class}}{\text{total \# of words in docs from this class}}$$

- Maximum likelihood from the training data

$$\prod_{d=1}^{D} \prod_{i=1}^{n_d} P(w_{d,i} \mid class_{d,i})$$

d: index of training document, *i*: index of a word

Parameter Estimation (3)

- The parameters of the model include likelihoods P(word|class) and the prior probability P(class)
- Training data

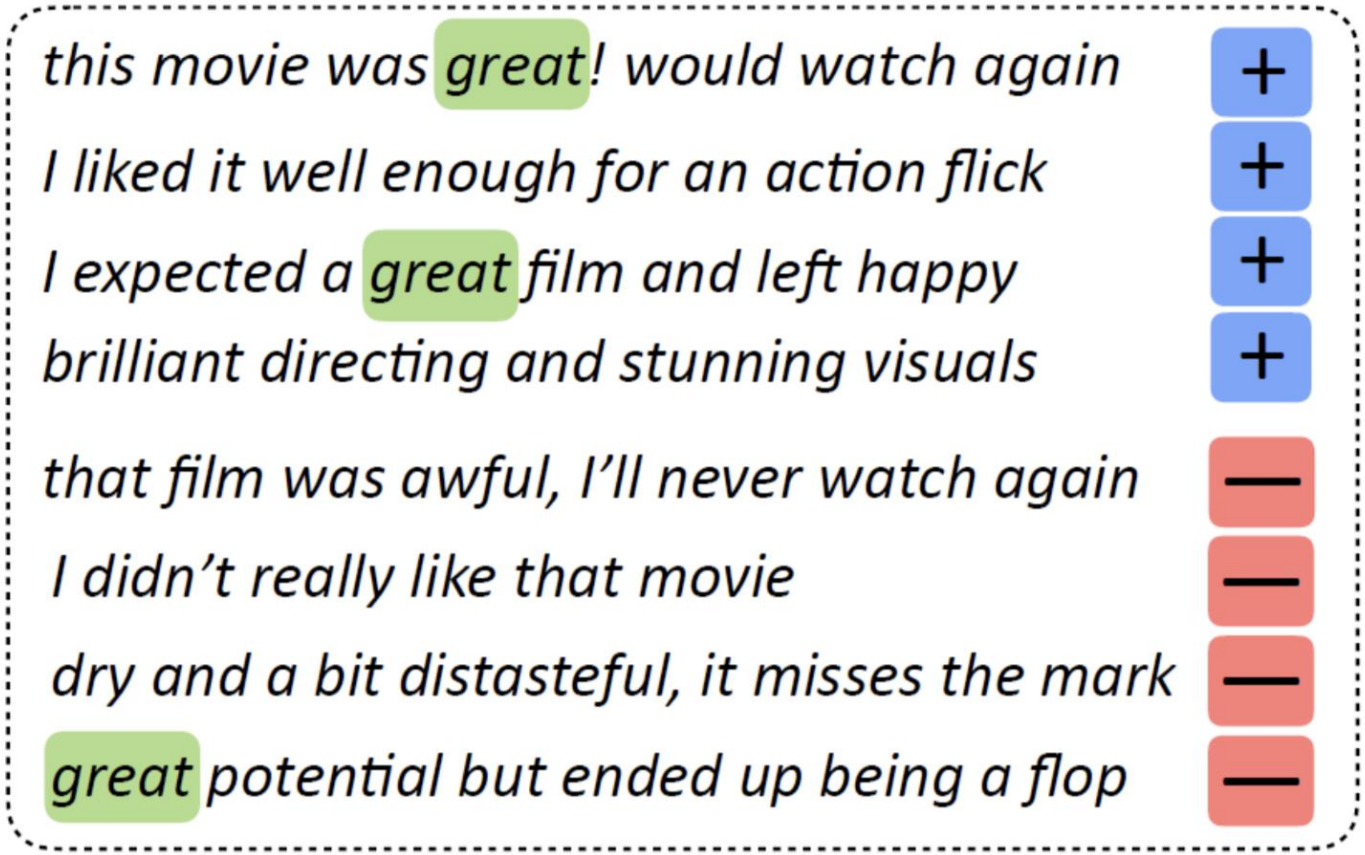
$$P(\text{word} \mid \text{class}) = \frac{\#\text{ of occurrences of this word in docs from this class}}{\text{total \# of words in docs from this class}}$$

- Parameter smoothing:
  - Laplace smoothing is a technique that helps tackle the problem of zero probability in the Naïve Bayes algorithm

$$P(\text{word} \mid \text{class}) = \frac{\#\text{ of occurrences of this word in docs from this class} + 1}{\text{total \# of words in docs from this class} + V}$$

(V: total number of unique words)

Example (1)



- Ex: It was great?

Example (2)

it was great $\longrightarrow$ $P(y|x) \propto \left[\quad \right]$

$$P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$\text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$$

Summary of Naïve Bayes Model

- Model

$$P(x, y) = P(y) \prod_{i=1}^{n} P(x_i|y)$$

- Inference

$$\text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$$

- Learning: maximize P(x,y) by counting data samples

Some problems of Naïve Bayes

the film was beautiful, stunning cinematography and gorgeous sets, but boring —

$$P(x_{\text{beautiful}}|+) = 0.1 \qquad P(x_{\text{beautiful}}|-) = 0.01$$
$$P(x_{\text{stunning}}|+) = 0.1 \qquad P(x_{\text{stunning}}|-) = 0.01$$
$$P(x_{\text{gorgeous}}|+) = 0.1 \qquad P(x_{\text{gorgeous}}|-) = 0.01$$
$$P(x_{\text{boring}}|+) = 0.01 \qquad P(x_{\text{boring}}|-) = 0.1$$

- Independent assumption
  - While “beautiful” and “gorgeous” are dependent
- Simple assumption
  - Computing P(x,y) via P(y|x)
- Discriminative model to compute P(y|x)

Application: Spam Filtering

Using Naïve Bayes Method

What is a spam message?

- An ads message which is sent automatically by a system to many people
- Un-expected messages

➔ Should be filtered

Spam Filtering Problem

- A document classification problem
- Binary classification
  - Spam
  - Not-Spam
- Mail corpus: Ling Spam & Enron Spam Corpus

Spam Data

- Data Corpus
  - 33,716 emails in 6 folders
  - Each folder contains two categories: spam / non-spam
  - 16,545 and 17,171 spam and non-spam messages, respectively

Email thường (ham)

Subject: re : indian springs this deal is to book the teco pvr revenue . it is my understanding that teco just sends us a check , i haven ' t received an answer as to whether there is a predermined price associated with this deal or if teco just lets us know what we are giving . i can continue to chase this deal down if you need .

Email rác (spam)

Subject: photoshop , windows , office . cheap . main trending abasements darer prudently fortuitous undergone lighthearted charm orinoco taster railroad affluent pornographic cuvier irvin parkhouse blameworthy chlorophyll robed diagrammatic fogarty clears bayda inconveniencing managing represented smartness hashish academies shareholders unload badness danielson pure caffein spaniard chargeable levin

Document Classification

- Input:
  - A document **d**
  - A set of classes: **C = {c1, c2, …}**
- Output:
  - Predict class **c** of document **d**

First Approach: Rules

- Extract features
- Build rules by combining features
  - For example: black-list-address OR (“dollars” AND “have been selected”)
- The accuracy may be very high if the rule-base is well designed
- However building and maintaining these rules is very time-consuming

Second Approach: Supervised Learning

- Input
  - A document d
  - A set of classes C
  - A collection of m labelled samples $(d_1, c_1),\ldots, (d_m,c_m)$
- Output
  - A classifier y: d➔c
- Methods: many supervised machine learning algorithms such as SVM, Decision Tree, Random Forest, Naïve Bayes

Practice

- Using the Ling Spam Corpus/Enron
- Pre-processing
- Feature Extraction
- Using Naïve Bayes and training/predicting

Pre-processing

- Applying pre-processing techniques in Lecture 3
- A document is represented by a sequence of tokens

```
Text: This is a cat. --> Word Sequence:
[this, is, a, cat]
```

- Applying regular expression to filter noises

Regular Expression

SYMBOL	USAGE
$	Matches the end of the line
\s	Matches whitespace
\S	Matches any non-whitespace character
*	Repeats a character zero or more times
\S	Matches any non-whitespace character
***?**	Repeats a character zero or more times (non-greedy)
+	Repeats a character one or more times
+?	Repeats a character one or more times (non-greedy)
[aeiou]	Matches a single character in the listed set
[^XYZ]	Matches a single character not in the listed set
[a-z0-9]	The set of characters can include a range

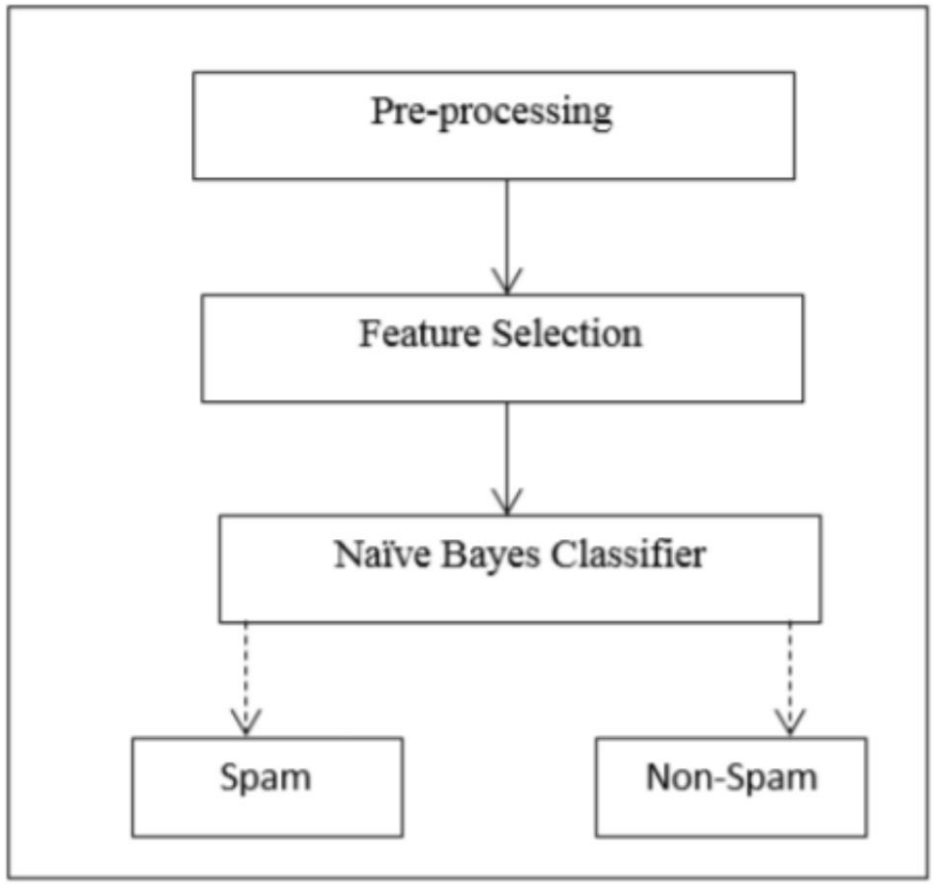
Example

- Remove number from the text
  - “My 2 favorite numbers are 8 and 25. My mobile number is 0912345678”
- Extract email from the text
  - “Hello from shubhamg199630@gmail.com to priya@yahoo.com about the meeting @2PM”

Feature Extraction

- Using Bag-of-words model as the simplest approach
- A vector BOW is high-dimensional (i.e., 80,000 dimensions)
- Using simple algorithms such as Naïve Bayes, linear SVM or logistic regression for high-dimensional feature vector

Classification Model



Improvements

- Using n-gram BOW
- Using TF-IDF
- Comparing different approaches for feature extraction

Homework

- Sentiment Analysis with data from Amazon
- http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Digital_Music_5.json.gz