



# Natural Language Processing



# Natural Language Processing

## **Text Classification**

Instructor: Moushmi Dasgupta

Connect up with me on LinkedIn  
[www.linkedin.com/in/moushmi1234](https://www.linkedin.com/in/moushmi1234)

# Session Plans

Date	Day	Topic	Time
2nd Nov 2022	Wednesday	Text Classification	9:30AM - 1:30PM
4th Nov 2022	Friday	Text Classification	3:00PM - 5:30PM
5th Nov 2022	Saturday	IE	3:00PM - 5:30PM
9th Nov 2022	Wednesday	IE	9:30AM - 1:30PM
11th Nov 2022	Friday	Chatbot	3:00PM - 5:30PM
13th Nov 2022	Saturday	Chatbot	3:00PM - 5:30PM

# AGENDA

## Text Classification

### Module 3

3.1 A Pipeline for Building Text Classification Systems

3.2 Using Existing Text Classification APIs

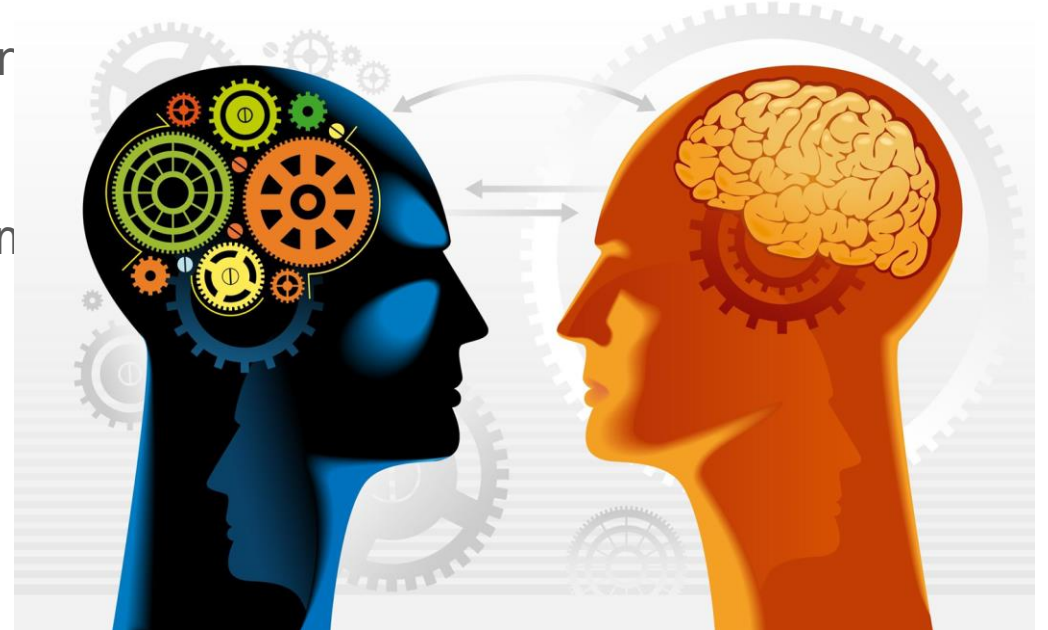
3.3 Use Case for Sentiment Analysis on Amazon Customer Reviews Data

### Pre-requisite:

1. Python programming
2. An understanding of Machine Learning
3. Fundamentals of NLP Understanding and hands-on with Python programming on and IDE
4. An Understanding of NLP Pipeline
5. Invest in attending classroom sessions (Weekly 1 or 2 classes of 3+ hours duration)
6. Invest in yourself with 1 hour of self study everyday

# Natural Language Processing

1. Natural Language Processing is a subfield of artificial intelligence concerned with methods of communication between computers and natural languages such as english, hindi, etc.
2. Objective of Natural Language processing is to perform useful tasks involving human languages like
  - Sentiment Analysis
  - Machine Translation
  - Part of Speech Tags
  - Human-Machine communication(chatbots)



# Why study NLP?

- ❑ Language is involved in most of the activities that involve interaction between humans, e.g. reading, writing, speaking, listening.
- ❑ Voice can be used as an interface for interactions between humans and machines e.g. cortana, google assistant, siri, amazon alexa.
- ❑ There is massive amount of data available in text format which can be used to derive insights from using NLP, e.g. blogs, research articles, consumer reviews, literature, discussion forums.



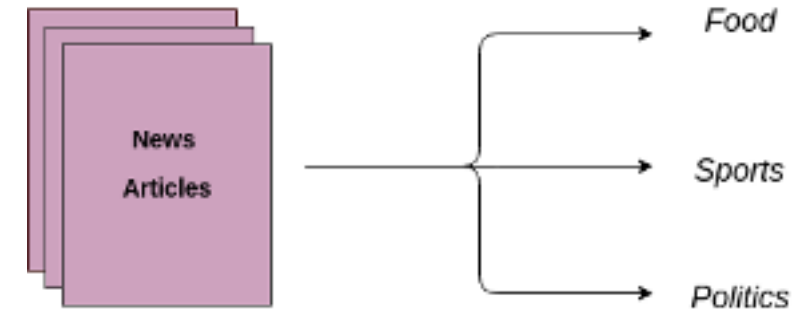
# Different Tasks in NLP

- **Text Classification**

- Sentiment Analysis: Determining the general context of a review, whether it is positive or negative or neutral.
- Consumer Complaints Classification: Categorizing complaints on consumer forums to respective departments.

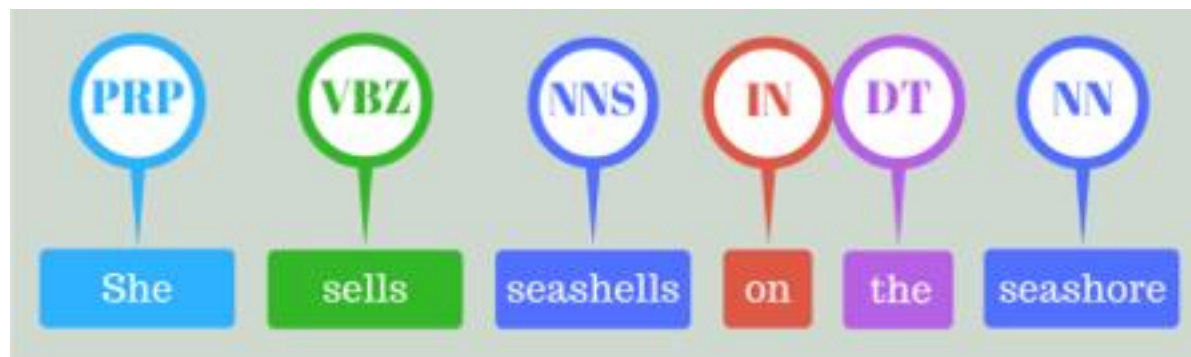
- **Machine Translation**

- Improving human-human interaction by translating sentences from one language to another.



# Different Tasks in NLP

- **Part of Speech Tagging**
  - In corpus linguistics, part-of-speech tagging (POS tagging or PoS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context.
  - A simplified form of this is the identification of words as nouns, verbs, adjectives, adverbs, etc.
  - Tagset: [https://www.ling.upenn.edu/courses/Fall\\_2003/ling001/penn\\_treebank\\_pos.html](https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)





# Movie Ratings

positive

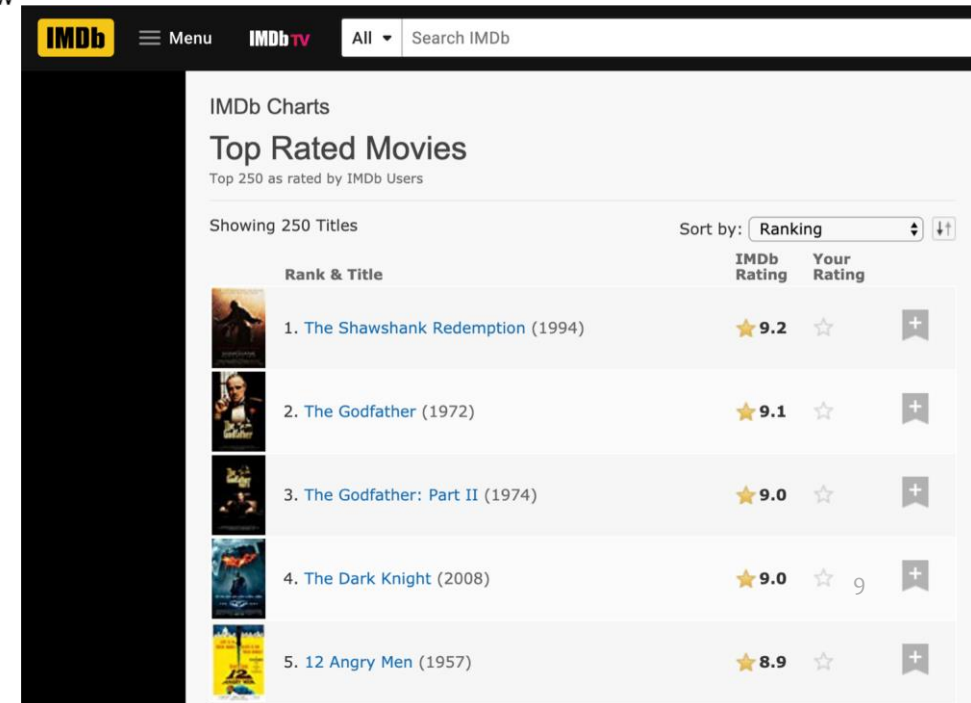
“... is a film which still causes real, not figurative, chills to run along my spine, and it is certainly the bravest and most ambitious fruit of Coppola's genius”

Roger Ebert, Apocalypse Now

- “I hated this movie. Hated hated hated hated hated this movie. Hated it. Hated every simpering stupid vacant audience-insulting moment of it. Hated the sensibility that thought anyone would like it.”

Roger Ebert, North

negative

A screenshot of the IMDb website showing the 'Top Rated Movies' chart. The page header includes the IMDb logo, a menu icon, 'IMDb TV', and a search bar. The main content area is titled 'IMDb Charts' and 'Top Rated Movies', with a subtitle 'Top 250 as rated by IMDb Users'. It indicates 'Showing 250 Titles' and a 'Sort by: Ranking' dropdown. The table lists the top 5 movies with their rank, title, year, IMDb rating (stars), and a 'Your Rating' column with a star icon and a plus button.

Rank & Title	IMDb Rating	Your Rating
1. <a href="#">The Shawshank Redemption</a> (1994)	★ 9.2	☆ +
2. <a href="#">The Godfather</a> (1972)	★ 9.1	☆ +
3. <a href="#">The Godfather: Part II</a> (1974)	★ 9.0	☆ +
4. <a href="#">The Dark Knight</a> (2008)	★ 9.0	☆ 9 +
5. <a href="#">12 Angry Men</a> (1957)	★ 8.9	☆ +

# Customer Review



## NOT DISHWASHER SAFE

Reviewed in the United States on April 5, 2019

Color: Blue | **Verified Purchase**

Used the bottle for one day. There was a slight lid leak, but I was willing to overlook that because I liked the other aspects of the product. Put it in the dishwasher with my other water bottles, air dry, and it melted. There is nothing in the product description that indicates it is not dishwasher safe, nor was there a product sheet included with the bottle indicating to hand wash only. I have a number of plastic water bottles that I routinely send through the dishwasher on this setting and have never had a problem. Extremely disappointed!

19 people found this helpful

Helpful

Comment

Report abuse



## Makes Drinking Water Fun

Reviewed in the United States on March 31, 2019

Color: Transparent | **Verified Purchase**

It is always a challenge to drink the recommended amount of water each day, so important for health. This bottle makes it fun while serving as a reminder to keep drinking! Bottle is good quality, handle makes it easy to lift.



14 people found this helpful

## Customer reviews



4.5 out of 5

451 customer ratings



## By feature

Sturdiness	<div><div></div></div> 4.5
Flavor	<div><div></div></div> 4.5
Durability	<div><div></div></div> 4.4

# Political Opinion Mining



**emilia** @PoliticalEmilia · 43m

As somebody whose immediate family are **immigrants** from Iran, I want to remind that this isn't the fault of Iranian Americans. Most of us want no more war in the Middle East.

Take your anger out at your government leaders, not at us. We have nothing to do with it. [#IranAttacks](#)

81

239

1.9K



**Nithya Raman** @nithyavraman · Jan 6

LA is one of the most **immigrant**-rich cities in the US.

Almost 50% of residents are foreign-born. 10% are undocumented.

As Trump works to implement his racist agenda, what are our elected officials doing to defend **immigrant** Angelenos?

The answer: infuriatingly little. (thread)

55

138

606



**Brigitte Gabriel** @ACTBrigitte · 3m

Thank Goodness there were ZERO U.S. casualties from the attacks Iran made tonight.

President **Trump** is monitoring the situation with his top leaders right now.

I've never felt more comfortable with a leader at the helm, than I do tonight with President **Trump** in office.

21

145

413



**Palmer Report** @PalmerReport · 1m

So a foreign nation fired missiles at U.S. troops tonight, and the President of the United States ISN'T addressing the nation? How far gone is Donald **Trump**? His handlers don't even trust him to read a speech off a teleprompter anymore.

15

74

225



**Andrea Chalupa** @AndreaChalupa · 7m

**Trump** is betting on Iran doing something so horrific to Americans that we rally around the flag, and the 2020 election becomes a mindless debate of who's "patriotic" vs. who's anti-war ("weak" on Iran).

47

147

425



# Some Direct Text Classification Applications

Task	$x$	$y$
Language identification	text	{English, Mandarin, Greek, ...}
Authorship attribution	text	{jk rowling, james joyce, ...}
Sentiment classification	text	{positive, negative, neutral, mixed}

# Summary of Linear Classification

	<b>Pros</b>	<b>Cons</b>
Naive Bayes	Simple, probabilistic, fast Closed-form	Not very accurate
Logistic Regression	Error-driven learning, regularized	More difficult to implement

# Different Tasks in NLP

- **Word Segmentation**
  - In some languages, there is no space between words, or a word may contain smaller syllables. In such languages, word segmentation is the first step of NLP systems.
- **Semantic Analysis**
  - Semantic analysis of a corpus (a large and structured set of texts) is the task of building structures that approximate concepts from a large set of documents.
  - Application of Semantic Analysis:
    - Text Similarity
    - Context Recognition
    - Sentence Parsing
    - Topic Modelling

# Why NLP is hard?

- ❑ Languages are changing everyday, new words, new rules, etc.
- ❑ The number of tokens is not fixed. A natural language can have hundreds of thousands of different words, new words are created on the fly.
- ❑ Words can have different meanings depending on context, and they can acquire new meanings over time (apple(a fruit), Apple(the company)], they can even change their parts of speech(Google --> to google).
- ❑ Every language has its own uniqueness.
- ❑ Like in the case of English we have words, sentences, paragraphs and so on to limit our language. But in Thai, there is no concept of sentences.



# Pre-processing Steps



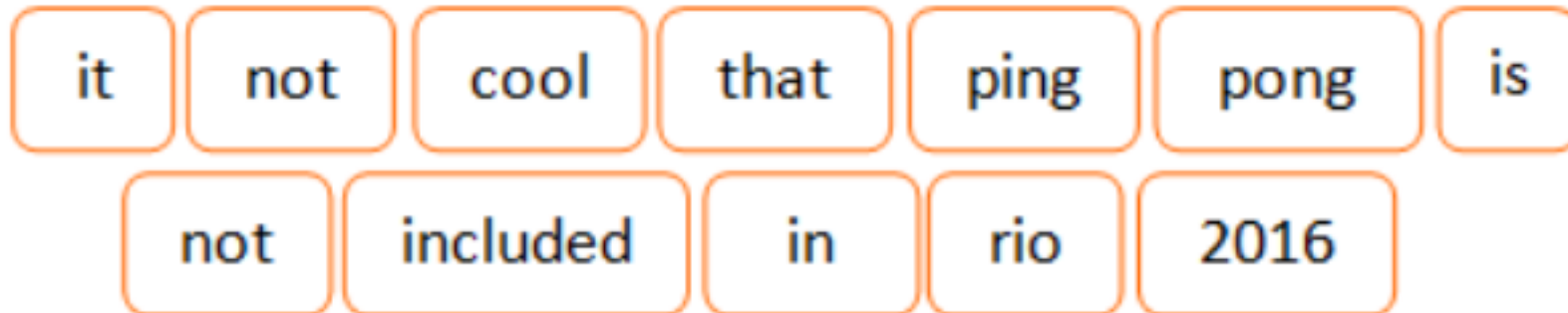
# Tokenization

- Tokenization is the task of taking a text or set of text and breaking it up into its individual tokens.
- Tokens are usually individual words (at least in languages like English).
- Tokenization can be achieved using different methods. Most common method is Whitespace tokenizer and Regexp Tokenizer. We will use them in our case study.

it not cool that ping pong is not included in rio 2016

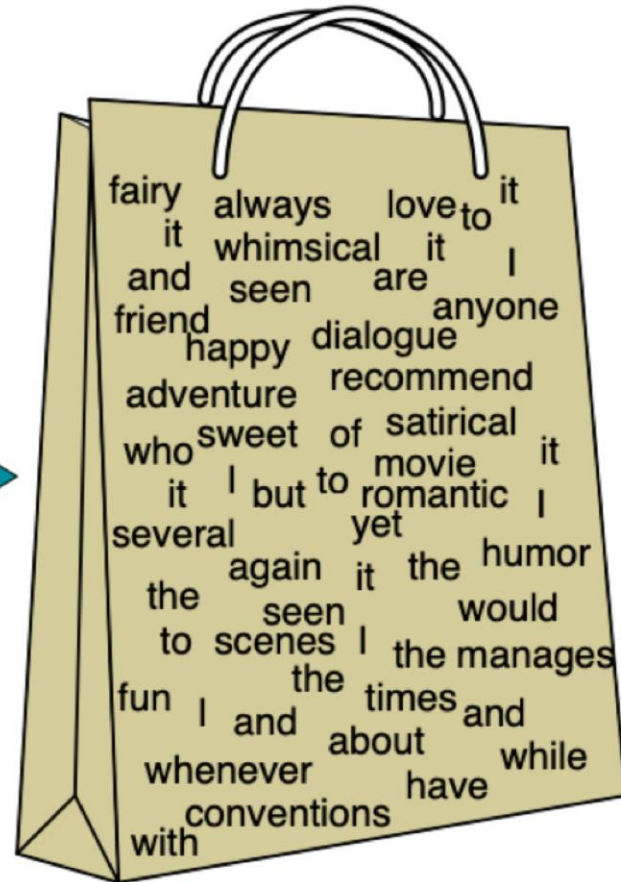


**Tokenization**



# Bag-of-Words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

# The Bag-of-Words

- One challenge is that the sequential representation  $(w_1, w_2, \dots, w_T)$  may have a different length  $T$  for every document.

- The bag-of-words is a fixed-length representation, which consists of a vector of word counts:

$\mathbf{w}$  = It was the best of times, it was the worst of times

$\mathbf{x} = [\underbrace{\text{aardvark}}_0, \dots, \underbrace{\text{best}}_1, \dots, \underbrace{\text{it}}_2, \dots, \underbrace{\text{of}}_2, \dots, \underbrace{\text{zyther}}_0]$

- The length of  $\mathbf{x}$  is equal to the size of the vocabulary  $V$
- For each  $\mathbf{x}$ , there may be many possible  $\mathbf{w}$ , depending on word order.

# Stop Words Removal

- Stopwords are common words that carry less important meaning than keywords.
- When using some bag of words based methods, i.e, countVectorizer or tfidf that works on counts and frequency of the words, removing stopwords is great as it lowers the dimensional space.
- Not always a good idea?
  - When working on problems where contextual information is important like machine translation, removing stop words is not recommended.

```
> stopwords("english")
[1] "i"      "me"      "my"      "myself"  "we"
[6] "our"    "ours"    "ourselves" "you"     "your"
[11] "yours"  "yourself" "yourselves" "he"      "him"
[16] "his"    "himself" "she"      "her"     "hers"
[21] "herself" "it"      "its"      "itself"  "they"
[26] "them"   "their"   "theirs"   "themselves" "what"
[31] "which"  "who"     "whom"    "this"    "that"
[36] "these"  "those"   "am"      "is"      "are"
[41] "was"    "were"    "be"      "been"    "being"
[46] "have"   "has"     "had"     "having"  "do"
[51] "does"   "did"     "doing"   "would"   "should"
[56] "could"  "ought"   "i'm"     "you're"  "he's"
[61] "she's"  "it's"    "we're"   "they're" "i've"
[66] "you've" "we've"   "they've" "i'd"     "you'd"
[71] "he'd"   "she'd"   "we'd"    "they'd"  "i'll"
[76] "you'll" "he'll"   "she'll"  "we'll"   "they'll"
[81] "isn't"  "aren't"  "wasn't"  "weren't" "hasn't"
[86] "haven't" "hadn't"  "doesn't" "don't"   "didn't"
[91] "won't"   "wouldn't" "shan't"  "shouldn't" "can't"
[96] "cannot"  "couldn't" "mustn't" "let's"   "that's"
[101] "who's"   "what's"  "here's"  "there's" "when's"
[106] "where's" "why's"   "how's"   "a"       "an"
```

# Stemming and Lemmatization

- ❑ The idea of reducing different forms of a word to a core root.
- ❑ Words that are derived from one another can be mapped to a central word or symbol, especially if they have the same core meaning.
- ❑ In stemming, words are reduced to their word stems. A word stem is an equal to or smaller form of the word.
- ❑ “cook,” “cooking,” and “cooked” all are reduced to same stem of “cook.”
- ❑ Lemmatization involves resolving words to their dictionary form. A lemma of a word is its dictionary or canonical form!



# Word Features

# Bag of Words

- In this model, a text (such as a sentence or a document) is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity.
- We use the tokenized words for each observation and find out the frequency of each token.
- We define the vocabulary of corpus as all the unique words in the corpus above and below some certain threshold of frequency.
- Each sentence or document is defined by a vector of same dimension as vocabulary containing the frequency of each word of the vocabulary in the sentence.
- The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

## Raw Text

it is a puppy and it  
is extremely cute

## Bag-of-words vector

it	2
they	0
puppy	1
and	1
cat	0
aardvark	0
cute	1
extremely	1
...	...

# Tf-idf Vectors

- Tf-idf (term frequency times inverse document frequency) is a scheme to weight individual tokens.
- One of the advantage of tf-idf is reduce the impact of tokens that occur very frequently, hence offering little to none in terms of information.

*TFIDF score for term  $i$  in document  $j = TF(i, j) * IDF(i)$*

*where*

*IDF = Inverse Document Frequency*

*TF = Term Frequency*

$$TF(i, j) = \frac{\text{Term } i \text{ frequency in document } j}{\text{Total words in document } j}$$

$$IDF(i) = \log_2 \left( \frac{\text{Total documents}}{\text{documents with term } i} \right)$$

*and*

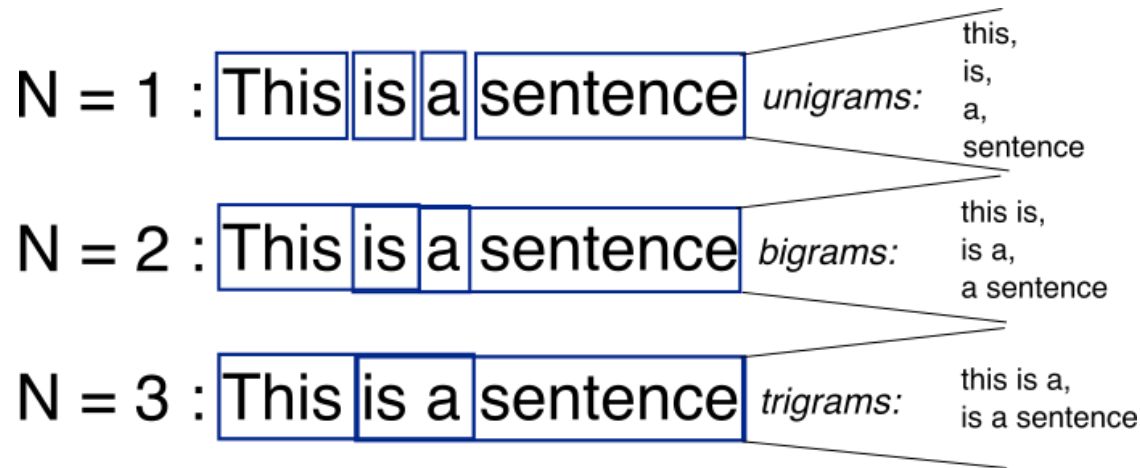
*$t$  = Term*

*$j$  = Document*



# N-gram and Language Model

- Language models are the type of models that assign probabilities to sequence of words.
- N-grams is the most simplest language model. It's a sequence of N-words.
- Bi-gram is a special case of N-grams where we consider only the sequence of two words (Markovian assumption ).
- In N-gram models we calculate the probability of Nth words give the sequence of N-1 words. We do this by calculating the relative frequency of the sequence occurring in the text corpus.



Bigram approximation

$$P(w_1^n) = \prod_{k=1}^n P(w_k | w_{k-1})$$

N-gram approximation

$$P(w_1^n) = \prod_{k=1}^n P(w_k | w_{k-N+1}^{k-1})$$

---

# NLP Pipeline Steps



Some of the material is from Georgia Institute of Technology, Atlanta, GA, USA.

# Data Acquisition

In the data acquisition step – lots of things to understand.

Data Availability – Where and how much is the Data

Data Acquisition Methods

- From Where
- How Much
- How much is enough
- Clean or not
- Sufficient or not
- Required or not
- Who has the data?
- How to get it?





# Text Cleaning / Pre-processing

1. Text Cleaning – HTML tag removing, Spelling checker, etc.
2. Basic Preprocessing —Tokenization(word or sent tokenization, stop word removal, removing digit, lower casing, etc
3. Advance Preprocessing — In this step we do POS tagging, Parsing, etc



# Lowercasing

As we know python is case sensitive language.

John, JOHN, john to work the same, converting to lower case is best.

Check out the script from NLTL to convert to lower case.

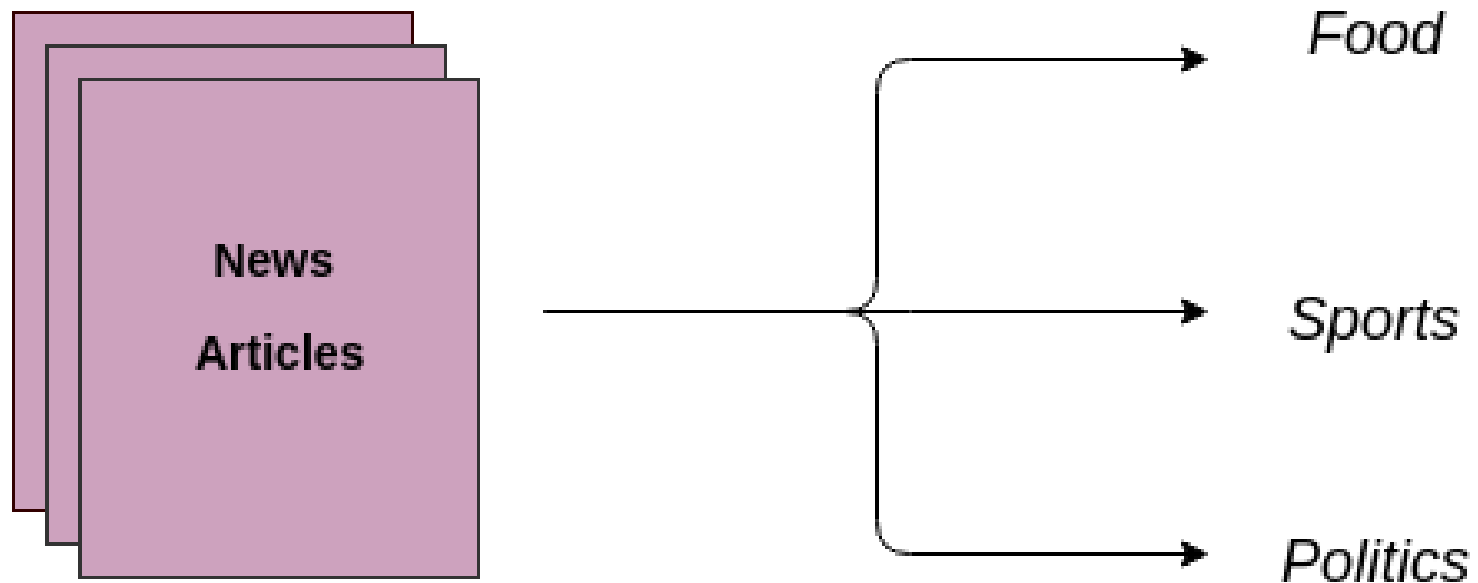
```
df['text'].str.lower()  
df['text'].apply(lambda  
x:x.lower())
```



# What's Next?

🧑 NLP Text Classification

# Text Classification



---

Text Classification is an example of supervised machine learning task since a labelled dataset containing text documents and their labels is used for train a classifier. An end-to-end text classification pipeline is composed of three main components:

**1. Dataset Preparation:** The first step is the Dataset Preparation step which includes the process of loading a dataset and performing basic pre-processing. The dataset is then splitted into train and validation sets.

**2. Feature Engineering:** The next step is the Feature Engineering in which the raw dataset is transformed into flat features which can be used in a machine learning model. This step also includes the process of creating new features from the existing data.

**3. Model Training:** The final step is the Model Building step in which a machine learning model is trained on a labelled dataset.

**4. Improve Performance of Text Classifier:** In this article, we will also look at the different ways to improve the performance of text classifiers.



---

# What is Text Classification actually?

Detecting patterns is a central part of Natural Language Processing.

Words ending in *-ed* tend to be past tense verbs

Frequent use of *will* is indicative of news text.

These observable patterns — word structure and word frequency — happen to correlate with particular aspects of meaning, such as tense and topic.

# 1 Supervised Classification Classification

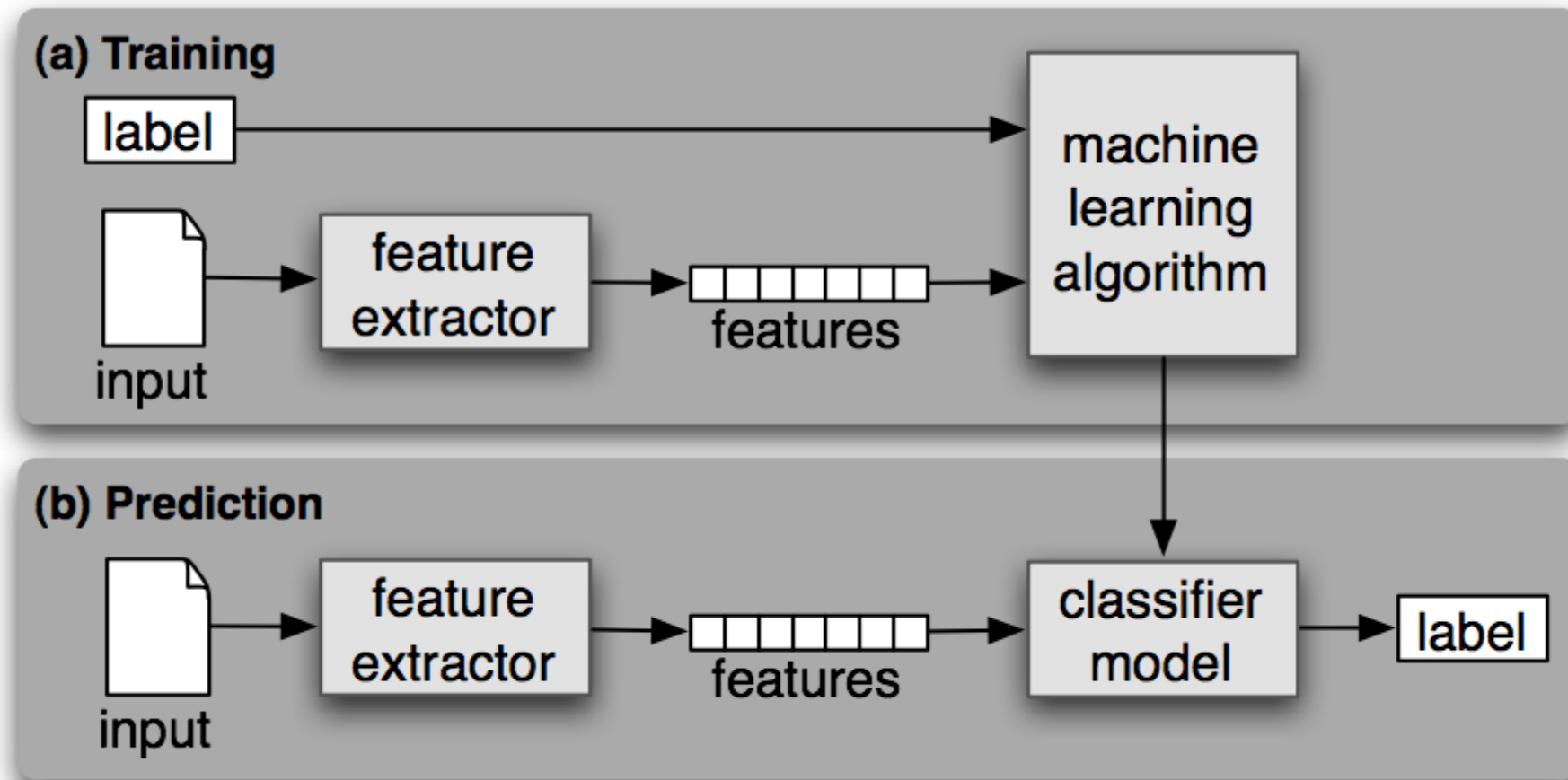
Task of choosing the correct **class label** for a given input.

In basic classification tasks, each input is considered in isolation from all other inputs, and the set of labels is defined in advance.

Some examples of classification tasks are:

- Deciding whether an email is spam or not.
- Deciding what the topic of a news article is, from a fixed list of topic areas such as "sports," "technology," and "politics."
- Deciding whether a given occurrence of the word *bank* is used to refer to a river bank, a financial institution, the act of tilting to the side, or the act of depositing something in a financial institution.

A classifier is called **supervised** if it is built based on training corpora containing the correct label for each input. The framework used by supervised classification is shown in





## **Gender Identification by the name given Name**

Check shared Python script and execute the exercises

## Choosing The Right Features

Check shared Python script and execute the exercises

Selecting relevant features and deciding how to encode them for a learning method can have an enormous impact on the learning method's ability to extract a good model.

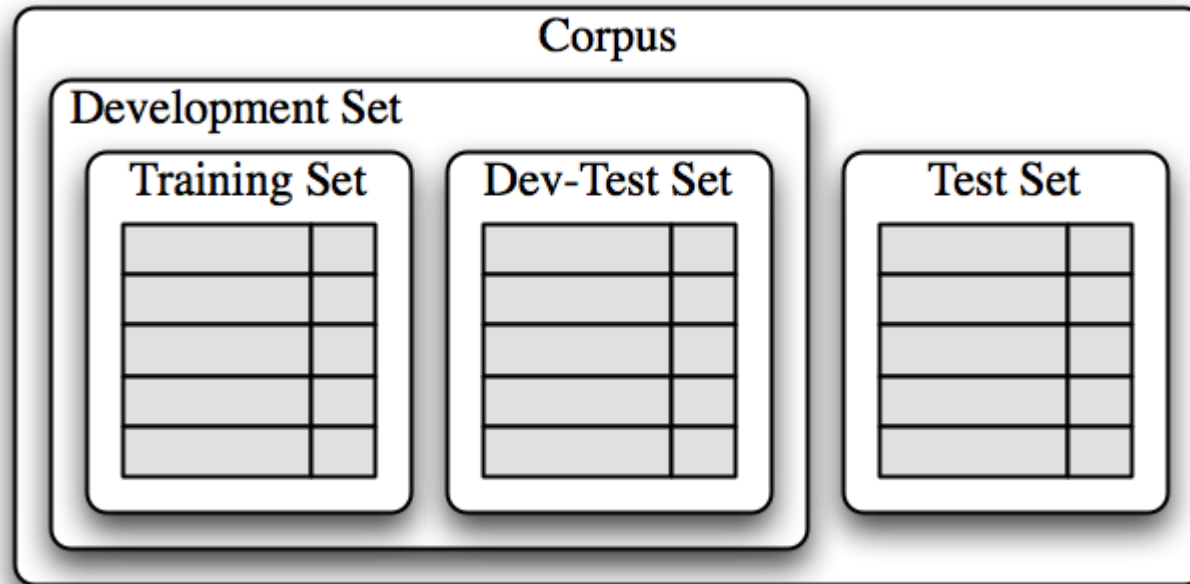
Much of the interesting work in building a classifier is deciding what features might be relevant, and how we can represent them.

Although it's often possible to get decent performance by using a fairly simple and obvious set of features, there are usually significant gains to be had by using carefully constructed features based on a thorough understanding of the task at hand.

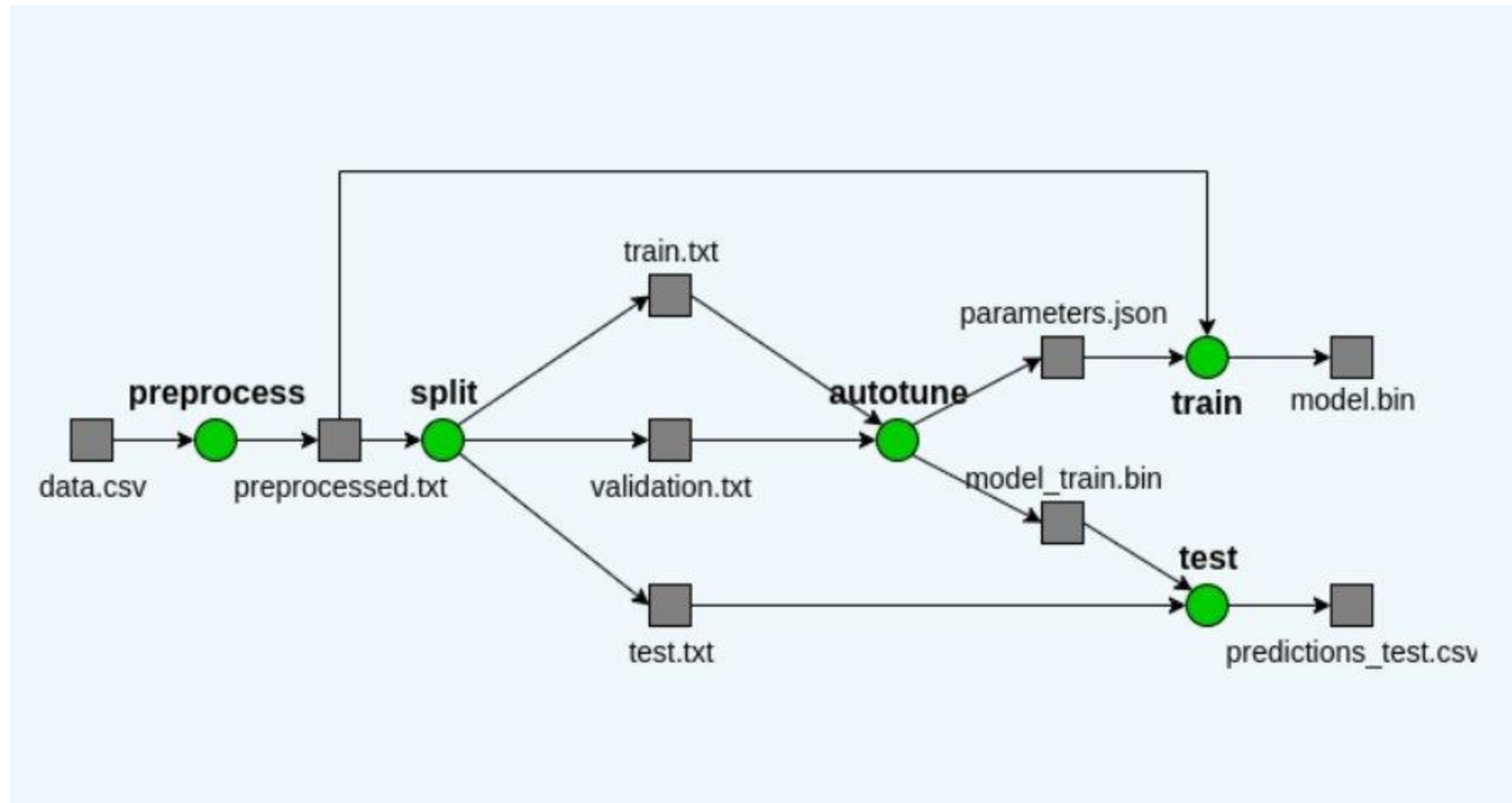
Typically, feature extractors are built through a process of **trial-and-error**, guided by intuitions about what information is relevant to the problem.

It's common to start with a "kitchen sink" approach, including all the features that you can think of, and then checking to see which features actually are helpful.

The training set is used to train the model, and the dev-test set is used to perform error analysis. The test set serves in our final evaluation of the system. For reasons discussed below, it is important that we employ a separate dev-test set for error analysis, rather than just using the test set.



### 3.1 A Pipeline for Building Text Classification Systems



A production level pipeline for building Txt classification systems

---

## 3.2 Using Existing Text Classification APIs

### **Open-Source Libraries for Text Classification**

1. Scikit-learn
2. Natural Language Toolkit (NLTK)
3. SpaCy

(Other Open source Libraries of interest)

1. TensorFlow
2. PyTorch
3. Keras

### **Paid API's**

Google Cloud NLP  
IBM Watson  
Lexalytics  
Amazon Comprehend  
Aylien



---

# Scikit Learn – Text Analytics

[https://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)

Practice labs as demoed in class

---

## NLTK – Natural Language Toolkit

<https://www.nltk.org/>

Perform all labs as demoed in the Class

---

# SpaCy

<https://spacy.io/>

Perform all labs as demoed in the Class

---

## Named Entity Recognition:

A named entity is a "real-world object" that's assigned a name. Example, a person, a country, a product or a book title.

We also get named entity recognition as part of spacy package.

It is inbuilt in the english language model and we can also train our own entities if needed.

---

# Dependency Parser

A dependency parser analyzes the grammatical structure of a sentence.

It establishing relationships between "head" words and words which modify those heads.

Spacy can be used to create these dependency parsers which can be used in a variety of tasks.

---

## Word Similarity:

Spacy has word vector model as well.  
We can use the same to find similar words.

---

## 3.3 Use Case for Sentiment Analysis on Amazon Customer Reviews Data

**Practical Case Study under Discussion**

Check and execute Scripts using NLP Tools and API



**More Material if required – for reference**

Some of the material is from Georgia Institute of Technology, Atlanta, GA, USA.



# Supervised Classification

## Given:

A description of an instance,  $d \in X$

$X$  is the *instance language* or *instance space*.

A fixed set of classes:

$$C = \{c_1, c_2, \dots, c_J\}$$

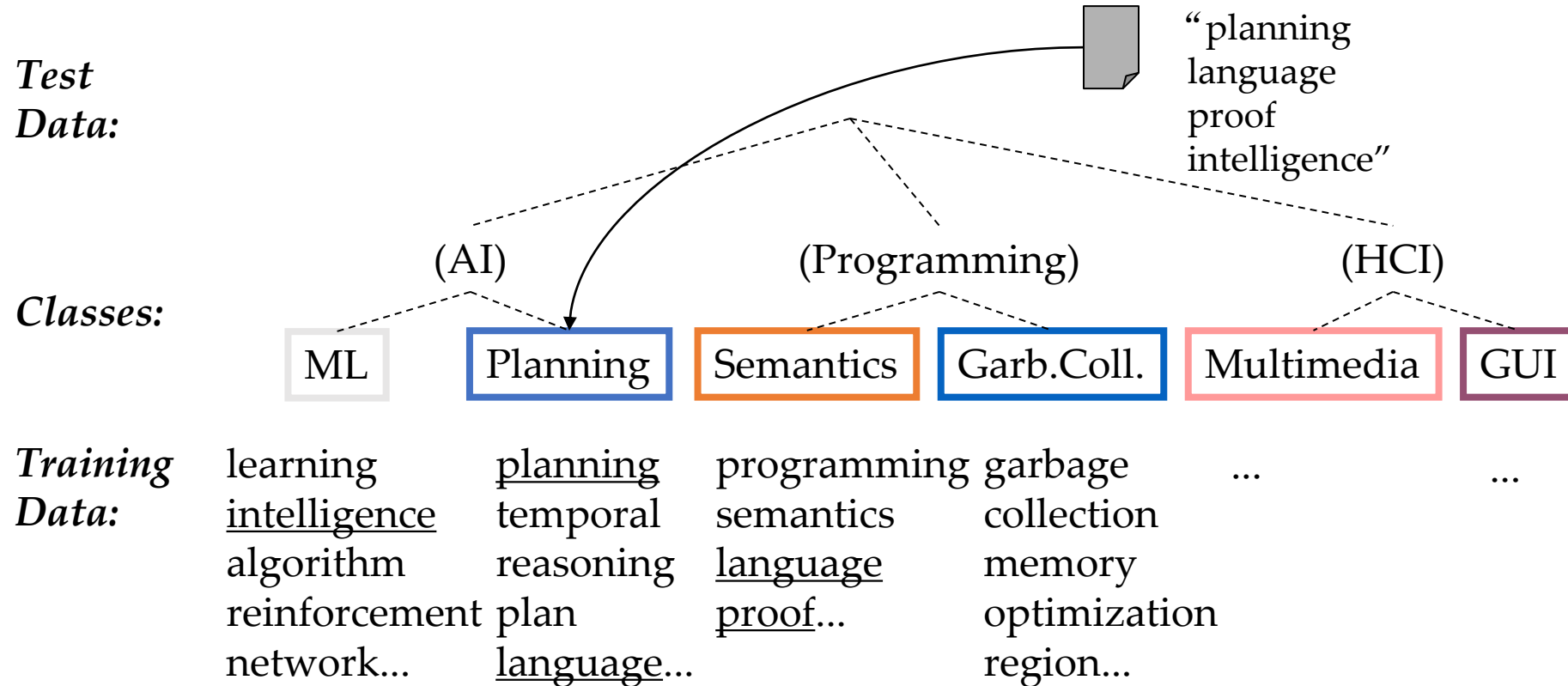
A training set  $D$  of labeled documents with each labeled document  $\langle d, c \rangle \in X \times C$

## Determine:

A learning method or algorithm which will enable us to learn a **classifier**  $\gamma: X \rightarrow C$

For a test document  $d$ , we assign it the class  $\gamma(d) \in C$

# Document Classification



(Note: in real life there is often a hierarchy, not present in the above problem statement; and also, you get papers on ML approaches to Garb. Coll.)

# More Text Classification Examples

Many search engine functionalities use classification

**Assigning labels to documents or web-pages:**

**Labels are most often topics such as Yahoo-categories**

*"finance," "sports," "news>world>asia>business"*

**Labels may be genres**

*"editorials" "movie-reviews" "news"*

**Labels may be opinion on a person/product**

*"like", "hate", "neutral"*

**Labels may be domain-specific**

*"interesting-to-me" : "not-interesting-to-me"*

*"contains adult language" : "doesn't"*

*language identification: English, French, Chinese, ...*

*search vertical: about Linux versus not*

*"link spam" : "not link spam"*



# Naïve Bayes methods

# Probabilistic relevance feedback

**Rather than reweighting in a vector space...**

**If user has told us some relevant and some irrelevant documents, then we can proceed to build a probabilistic classifier,**

such as the Naive Bayes model we will look at today:

$$P(t_k | R) = |\mathbf{D}_{rk}| / |\mathbf{D}_r|$$

$$P(t_k | NR) = |\mathbf{D}_{nrk}| / |\mathbf{D}_{nr}|$$

$t_k$  is a term;  $\mathbf{D}_r$  is the set of known relevant documents;  $\mathbf{D}_{rk}$  is the subset that contain  $t_k$ ;  $\mathbf{D}_{nr}$  is the set of known irrelevant documents;  $\mathbf{D}_{nrk}$  is the subset that contain  $t_k$ .

# Bayesian Methods

Learning and classification methods based on probability theory.

Bayes theorem plays a critical role in probabilistic learning and classification.

Builds a *generative model* that approximates how data is produced

Uses *prior* probability of each category given no information about an item.

Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

## Bayes' Rule for text classification

For a document  $d$  and a class  $c$

$$P(c, d) = P(c \mid d)P(d) = P(d \mid c)P(c)$$

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

# Naive Bayes Classifiers

**Task: Classify a new instance  $d$  based on a tuple of attribute values into one of the classes  $c_j \in C$**

$$d = \langle x_1, x_2, \dots, x_n \rangle$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j \mid x_1, x_2, \dots, x_n)$$

$$= \operatorname{argmax}_{c_j \in C} \frac{P(x_1, x_2, \dots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \dots, x_n)}$$

$$= \operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n \mid c_j) P(c_j)$$

MAP is “maximum a posteriori” = most likely class



# Naïve Bayes Classifier:

## Naïve Bayes Assumption

$$P(c_j)$$

Can be estimated from the frequency of classes in the training examples.

$$P(x_1, x_2, \dots, x_n | c_j)$$

$O(|X|^n \cdot |C|)$  parameters

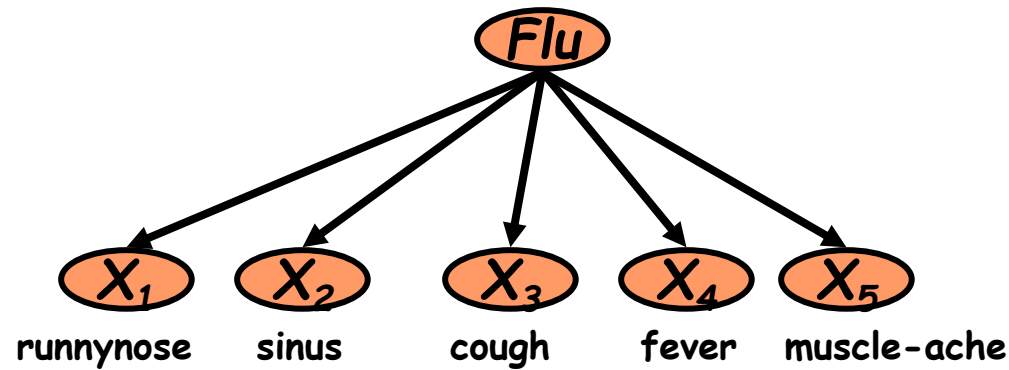
Could only be estimated if a very, very large number of training examples was available.

**Naïve Bayes Conditional Independence Assumption:**

**Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities**

$$P(x_i | c_j).$$

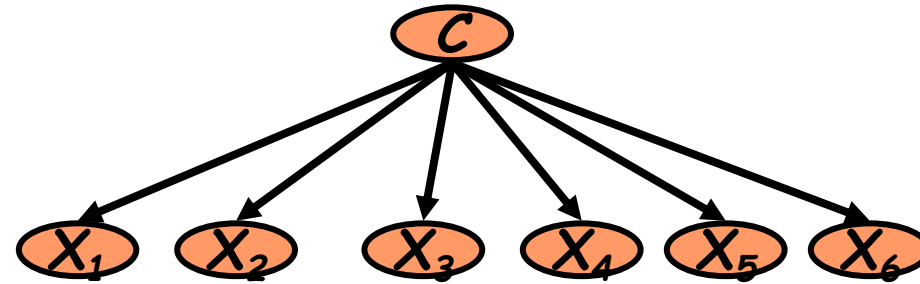
# The Naïve Bayes Classifier



**Conditional Independence Assumption:** features detect term presence and are independent of each other given the class:

$$P(X_1, \dots, X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \dots \bullet P(X_5 \mid C)$$

# Learning the Model



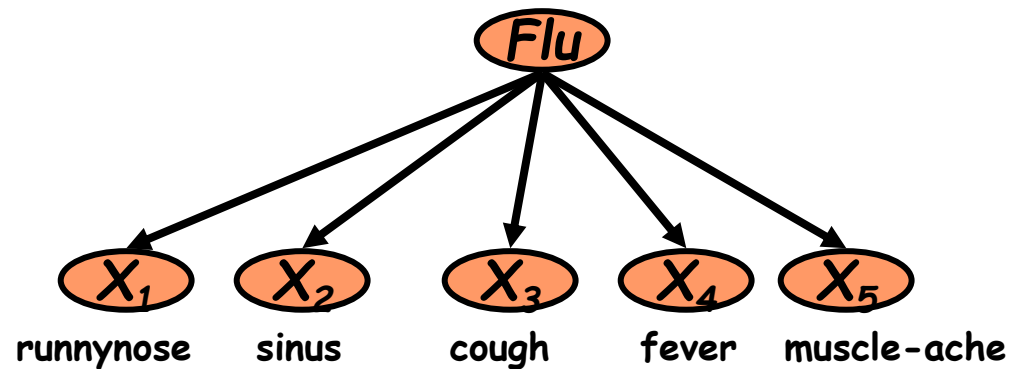
## First attempt: maximum likelihood estimates

simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

# Problem with Maximum Likelihood



$$P(X_1, \dots, X_5 | C) = P(X_1 | C) \bullet P(X_2 | C) \bullet \dots \bullet P(X_5 | C)$$

What if we have seen no training documents with the word *muscle-ache* and classified in the topic *Flu*?

$$\hat{P}(X_5 = t | C = f) = \frac{N(X_5 = t, C = f)}{N(C = f)} = 0$$

Zero probabilities cannot be conditioned away, no matter the other evidence!

Multivariate  
Bernoulli Model

$$\ell = \arg \max_c \hat{P}(c) \prod_i \hat{P}(x_i | c)$$

# Smoothing to Avoid Overfitting

$$\hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

# of values of  $X_i$

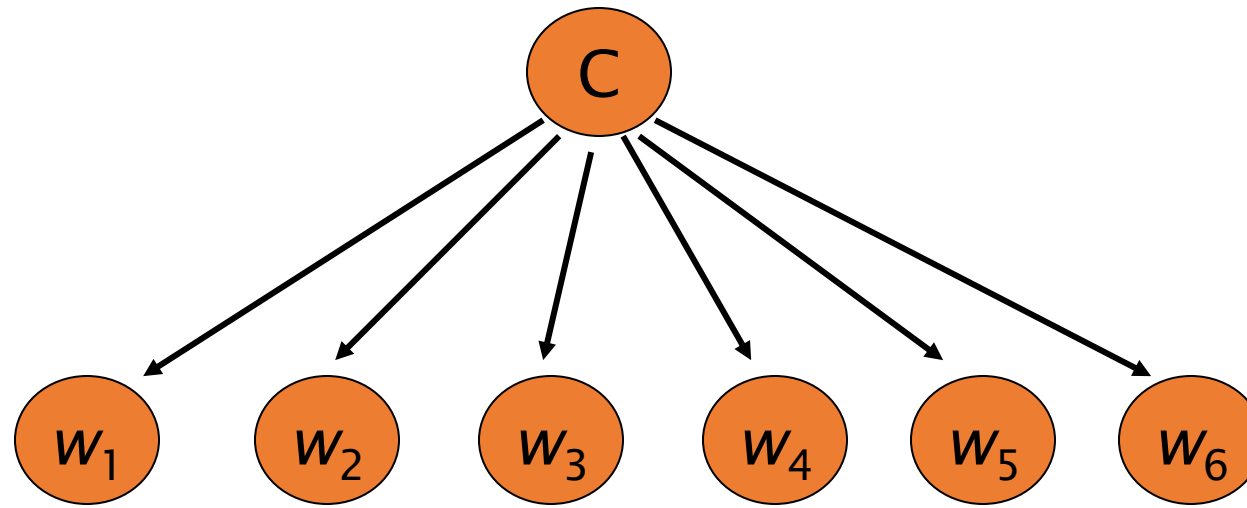
**Somewhat more subtle version**

overall fraction in  
data where  $X_i = x_{i,k}$

$$\hat{P}(x_{i,k} | c_j) = \frac{N(X_i = x_{i,k}, C = c_j) + mp_{i,k}}{N(C = c_j) + m}$$

extent of  
“smoothing”

Naïve Bayes via a class conditional language model  
= multinomial NB



**Effectively, the probability of each class is done as a class-specific unigram language model**

# Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

**Attributes are text positions, values are words.**

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j)$$

$$= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} | c_j) \cdots P(x_n = \text{"text"} | c_j)$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is bag-of-words model (over *tokens* not *types*)

# Naive Bayes: Learning

## Running example: document classification

**From training corpus, extract *Vocabulary***  
**Calculate required  $P(c_j)$  and  $P(x_k / c_j)$  terms**

For each  $c_j$  in  $C$  do

$docs_j \leftarrow$  subset of documents for which the target class is  $c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- $Text_j \leftarrow$  single document containing all  $docs_j$
- for each word  $x_k$  in *Vocabulary*
  - $n_{jk} \leftarrow$  number of occurrences of  $x_k$  in  $Text_j$
  - $n_j \leftarrow$  number of words in  $Text_j$

- $P(x_k | c_j) \leftarrow \frac{n_{jk} + 1}{n_j + |Vocabulary|}$



## Naive Bayes: Classifying

**positions**  $\leftarrow$  all word positions in current document  
which contain tokens found in *Vocabulary*

**Return**  $c_{NB}$ , where

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

# Naive Bayes: Time Complexity

**Training Time:  $O(|D|L_{ave} + |C||V|)$**   
**is the average length of a document in  $D$ .**

where  $L_{ave}$

Assumes all counts are pre-computed in  $O(|D|L_{ave})$  time during one pass through all of the data.

Generally just  $O(|D|L_{ave})$  since usually  $|C||V| < |D|L_{ave}$



**Test Time:  $O(|C| L_t)$**   
**the average length of a test document.**

where  $L_t$  is

**Very efficient overall, linearly proportional to the time needed to just read in all the data.**

# Underflow Prevention: using logs

Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.

Since  $\log(xy) = \log(x) + \log(y)$ , it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.

Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \operatorname{argmax}_{c_j \in C} [\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)]$$

Note that model is now just max of sum of weights...

# Naive Bayes Classifier

$$c_{NB} = \operatorname{argmax}_{c_j \in C} [\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)]$$

**Simple interpretation:** Each conditional parameter  $\log P(x_i | c_j)$  is a weight that indicates how good an indicator  $x_i$  is for  $c_j$ .

The prior  $\log P(c_j)$  is a weight that indicates the relative frequency of  $c_j$ .

The sum is then a measure of how much evidence there is for the document being in the class.

We select the class with the most evidence for it

# Two Naive Bayes Models

## Model 1: Multivariate Bernoulli

One feature  $X_w$  for each word in dictionary

$X_w = \text{true}$  in document  $d$  if  $w$  appears in  $d$

Naive Bayes assumption:

Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears

**This is the model used in the binary independence model in classic probabilistic relevance feedback on hand-classified data (Maron in IR was a very early user of NB)**

## Two Models

### Model 2: Multinomial = Class conditional unigram

One feature  $X_i$  for each word position in document  
feature's values are all words in dictionary

Value of  $X_i$  is the word in position  $i$

Naïve Bayes assumption:

Given the document's topic, word in one position in the document tells us  
nothing about words in other positions

Second assumption:

Word appearance does not depend on position

Just have one multinomial feature predicting all words

$$P(X_i = w | c) = P(X_j = w | c)$$

for all positions  $i, j$ , word  $w$ , and class  $c$

# Parameter estimation

## Multivariate Bernoulli model:

$$\hat{P}(X_w = t \mid c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$$

## Multinomial model:

$$\hat{P}(X_w = w \mid c_j) = \frac{\text{fraction of times in which word } w \text{ appears among all words in documents of topic } c_j}{\text{Can create a mega-document for topic } c_j \text{ by concatenating all documents in this topic}}$$

Use frequency of  $w$  in mega-document

# Classification

**Multinomial vs Multivariate Bernoulli?**

**Multinomial model is almost always more effective in text applications!**

See results figures later

**See *IIR* sections 13.2 and 13.3 for worked examples with each model**





# The rest of text classification methods

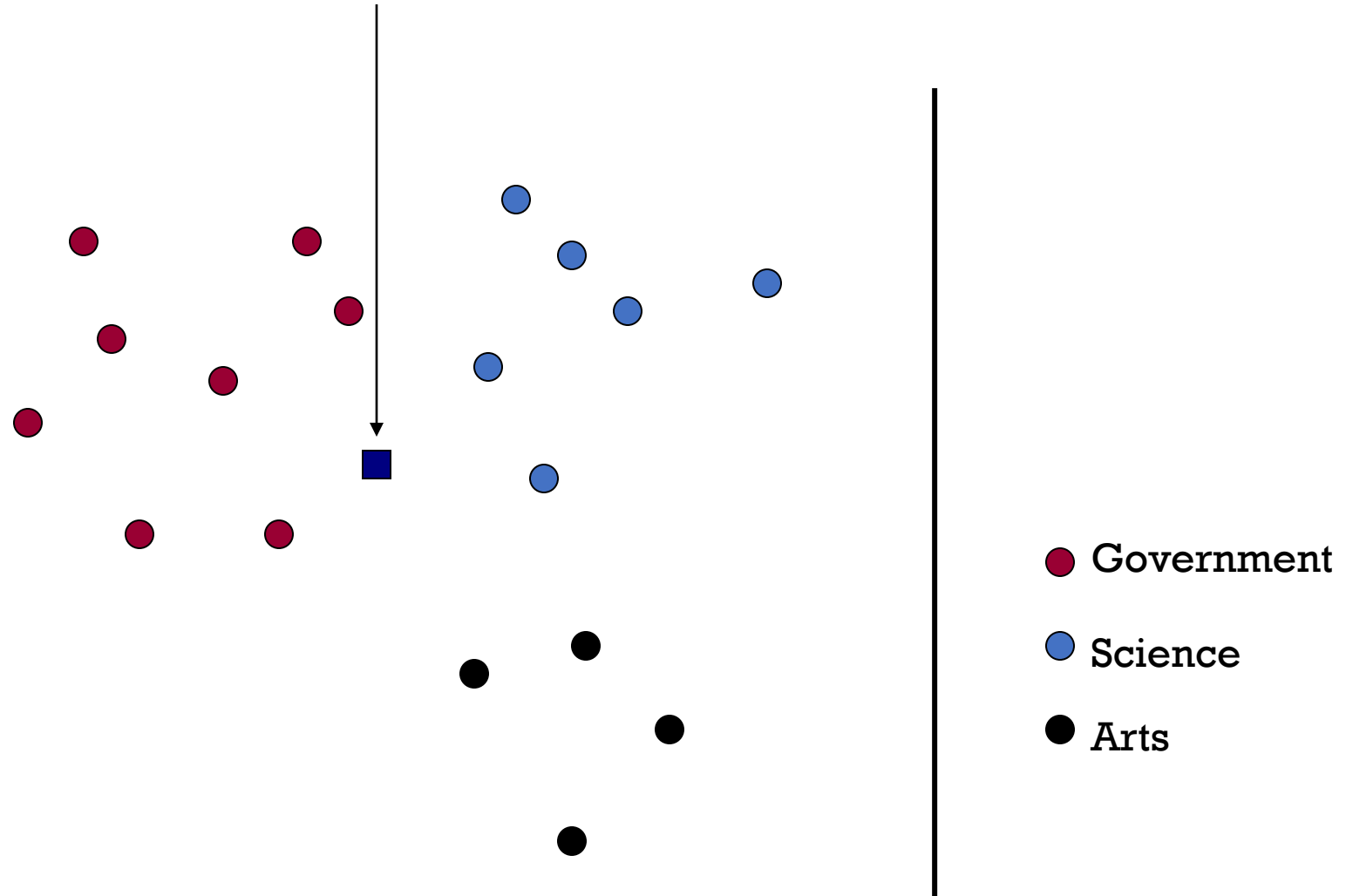
## **Vector space methods for Text Classification**

Vector space classification using centroids (Rocchio)

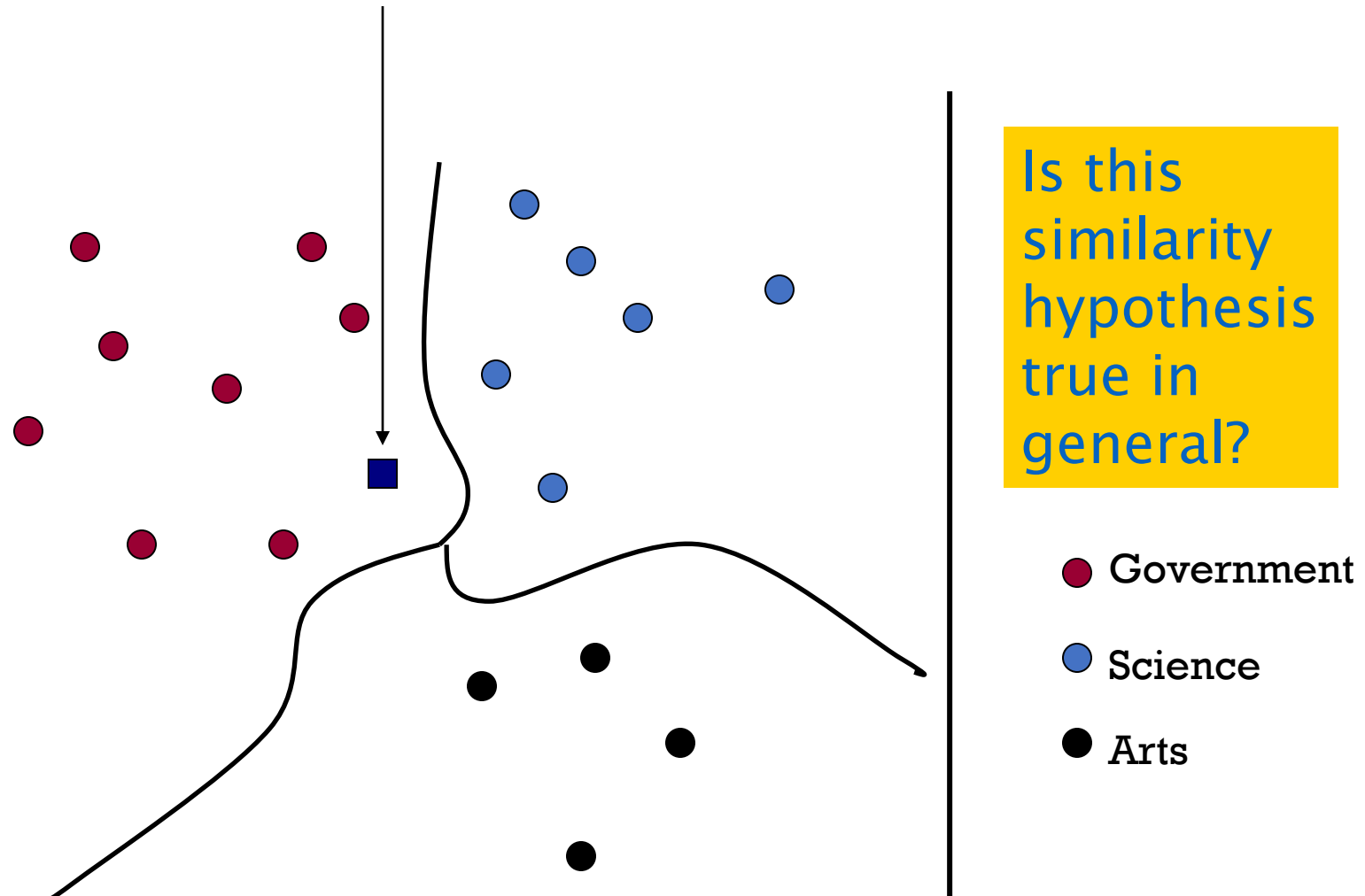
K Nearest Neighbors

Support Vector Machines

# Test Document of what class?



Test Document = Government



Our main topic today is how to find good separators



# kNN Classification

# k Nearest Neighbor Classification

**kNN = k Nearest Neighbor**

**To classify a document  $d$  into class  $c$ :**

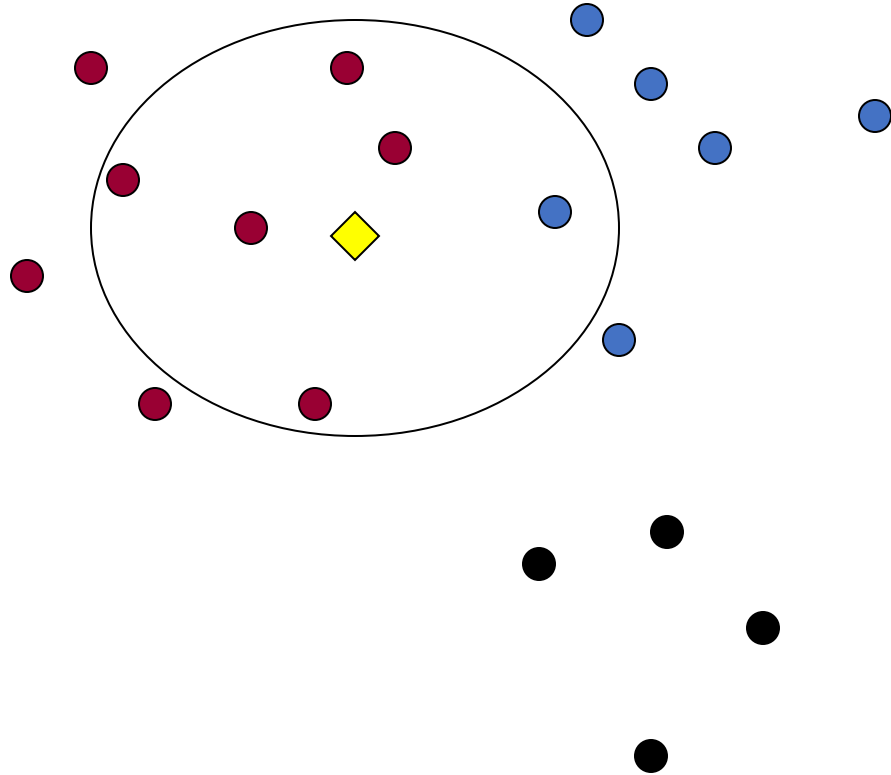
**Define  $k$ -neighborhood  $N$  as  $k$  nearest neighbors of  $d$**

**Count number of documents  $i_c$  in  $N$  that belong to  $c$**

**Estimate  $P(c|d)$  as  $i_c/k$**

**Choose as class  $\operatorname{argmax}_c P(c|d)$  [= majority class]**

## Example: k=6 (6NN)



$P(\text{science}|\text{diamond})?$

● Government

● Science

● Arts

# Nearest-Neighbor Learning Algorithm

**Learning is just storing the representations of the training examples in  $D$ .**

**Testing instance  $x$  (under 1NN):**

Compute similarity between  $x$  and all examples in  $D$ .

Assign  $x$  the category of the most similar example in  $D$ .

**Does not explicitly compute a generalization or category prototypes.**

**Also called:**

Case-based learning

Memory-based learning

Lazy learning

**Rationale of kNN: contiguity hypothesis**

# kNN Is Close to Optimal

**Cover and Hart (1967)**

**Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the Bayes rate** [error rate of classifier knowing model that generated data]

**In particular, asymptotic error rate is 0 if Bayes rate is 0.**

**Assume: query point coincides with a training point.**

**Both query point and training point contribute error  $\rightarrow$  2 times Bayes rate**



# k Nearest Neighbor

**Using only the closest example (1NN) to determine the class is subject to errors due to:**

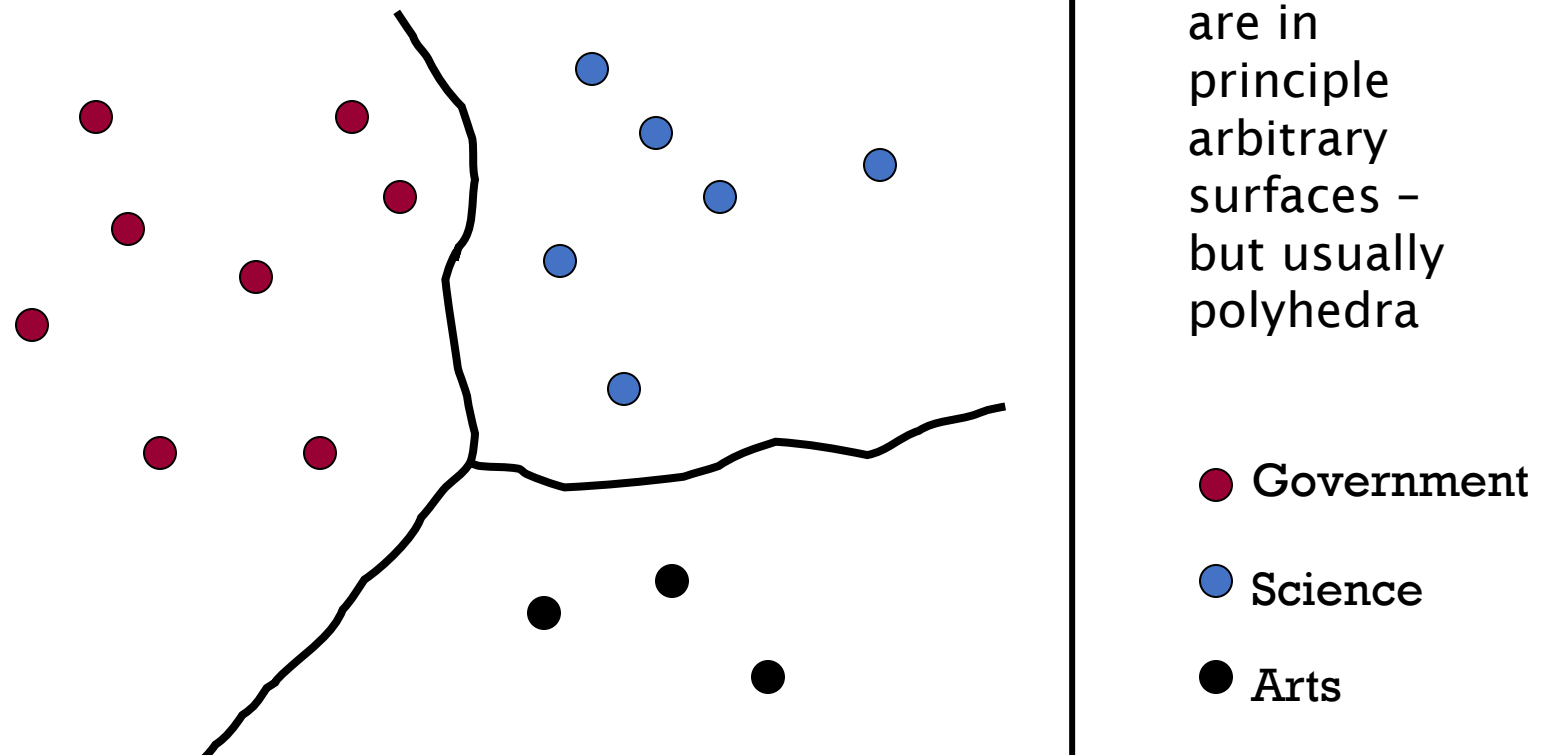
A single atypical example.

Noise (i.e., an error) in the category label of a single training example.

**More robust alternative is to find the  $k$  most-similar examples and return the majority category of these  $k$  examples.**

**Value of  $k$  is typically odd to avoid ties; 3 and 5 are most common.**

# kNN decision boundaries



kNN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, Rocchio, etc.)

## Similarity Metrics

Nearest neighbor method depends on a similarity (or distance) metric.

Simplest for continuous  $m$ -dimensional instance space is *Euclidean distance*.

Simplest for  $m$ -dimensional binary instance space is *Hamming distance* (number of feature values that differ).

For text, cosine similarity of tf.idf weighted vectors is typically most effective.