Introduction to Natural Language Processing

Introduction to Data Science 2017

Maria Veiga
PhD student @ Institute of Computational Science & Mathematics
mhanve@math.uzh.ch

What is NLP?

- Question answering

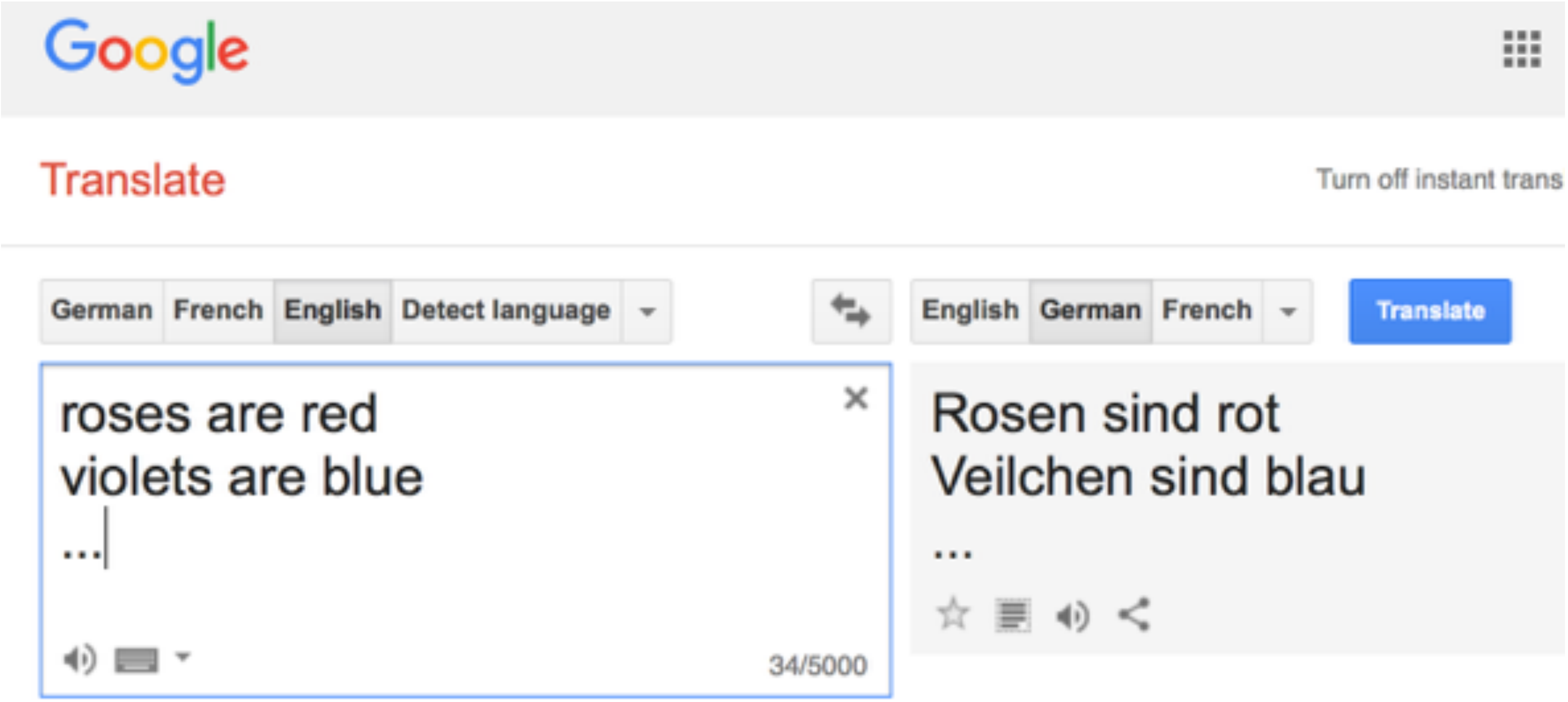
Computer Wins on 'Jeopardy!': Trivial, It's Not

Carol Kaelson/Jeopardy Productions Inc., via Associated Press

Two "Jeopardy!" champions, Ken Jennings, left, and Brad Rutter, competed against a computer named Watson, which proved adept at buzzing in quickly.

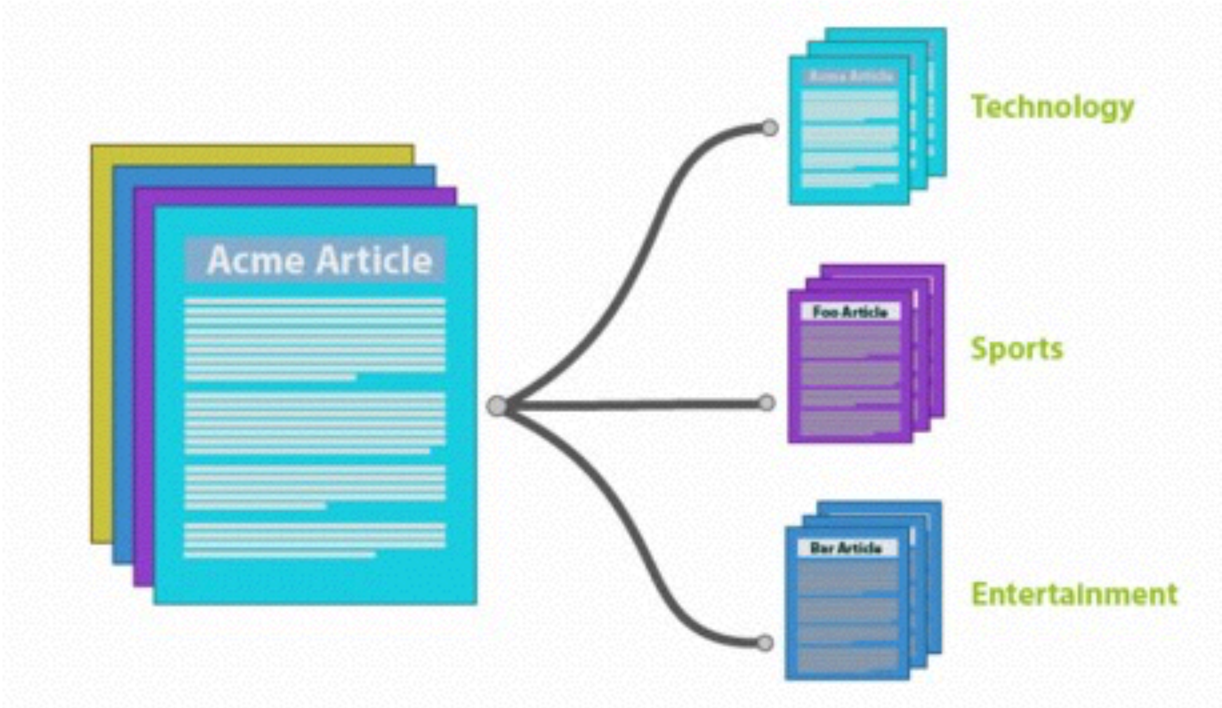
What is NLP?

- Machine translation



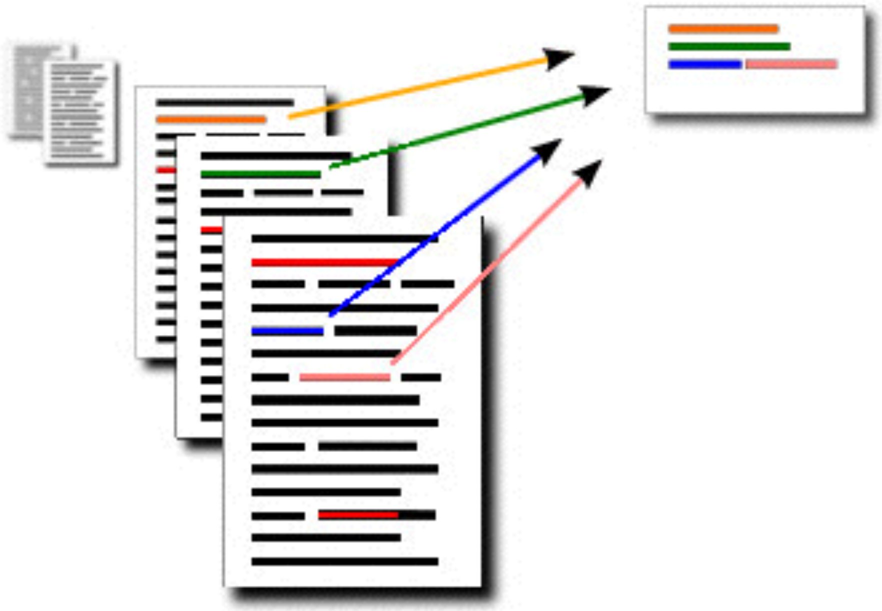
What is NLP?

- Document categorisation



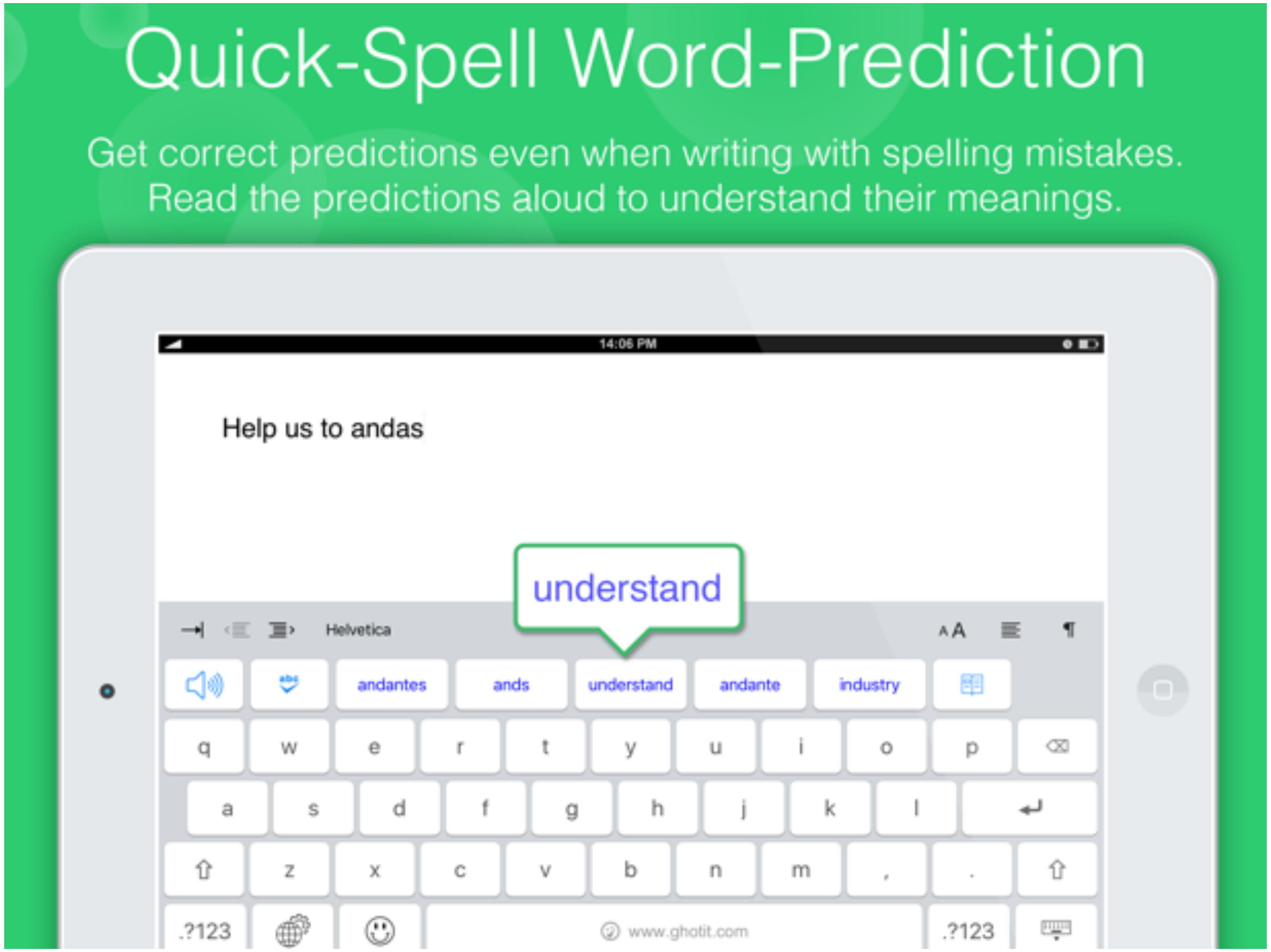
What is NLP?

- Document summarisation



What is NLP?

- Spelling correction



“Problem types”

- Supervised learning:
  - e.g: Document categorisation, sentiment analysis, spam detection, part-of-speech tagging
- Unsupervised learning:
  - e.g: Document clustering, summarization…
- Information retrieval:
  - e.g: Search engines, question answering…

Why is it hard?

"Money is coined liberty, and so it is ten times dearer to the man who is deprived of freedom. If money is jingling in his pocket, he is half consoled, even though he cannot spend it. But money can always and everywhere be spent, and, moreover, forbidden fruit is sweetest of all."

Why is it hard?

ambiguity
times - mathematical operation vs
plural of time

“Money is coined liberty, and so it is ten **times** dearer to the man who is deprived of freedom. If money is jingling in his pocket, he is half consoled, even though he cannot spend it. But money can always and everywhere be spent, and, moreover, forbidden fruit is sweetest of all.”

Why is it hard?

Synonyms

“Money is coined **liberty**, and so it is ten times dearer to the man who is deprived of **freedom**. If money is jingling in his pocket, he is half consoled, even though he cannot spend it. But money can always and everywhere be spent, and, moreover, forbidden fruit is sweetest of all.”

Why is it hard?

“Money is coined liberty, and so it is ten times dearer to the man who is deprived of freedom. If money is jingling in his pocket, he is half consoled, even though he **cannot** spend it. But money can always and everywhere be spent, and, moreover, forbidden fruit is sweetest of all.”

abbreviations/contractions
non-standardised text

Why is it hard?

“Money is coined liberty, and so it is ten times dearer to the man who is deprived of freedom. If money is jingling in his pocket, he is half consoled, even though he cannot spend it. But money can always and everywhere be spent, and, moreover, **forbidden fruit is sweetest of all**.”

idioms

Why is it hard?

“Money is coined liberty, and so it is ten times dearer to the man who is deprived of freedom. If money is jingling in his pocket, he is half consoled, even though he cannot spend it. But money can always and everywhere be spent, and, moreover, forbidden fruit is sweetest of all.”

How to represent text, segment, sequential dependence, etc.

Why is it hard?

- Ambiguity
- Synonyms
- Abbreviations, typos
- Idioms
- Word segmentation “New York” -> “New” “York” or “New York”

Outline

- Hour 1: Theory
  - Basic concepts
  - Feature representation
  - Text classification
- Hour 2: Practice
  - Text classification: example
  - Exercises (YOU CHOOSE):
    - Text classification
    - Text clustering

Basic concepts

Text normalisation:

- Tokenisation
- Normalisation
- Stemming/Lemmatisation

Tokenisation

Given a sentence/text, tokenisation is the task of chopping it up into pieces, called *tokens*, perhaps at the same time throwing away certain characters, such as punctuation.

e.g:

Input: “Friends, Romans, Countrymen, lend me your ears;”

Output: “Friends”, “Romans”, “Countrymen”, “lend”, “me”, “your”, “ears”

Normalisation

After tokenising a document/text, we might want some tokens to be “merged”

e.g:

“antidiscrimination” “anti-discrimination” -> “antidiscrimination”

“Today” “today” -> “today”

But this is not always obvious!

E.g.: “C.A.T” should not be mapped to “cat”

Stemming/Lemmatisation

It’s often to have different forms of a word in a document.

e.g: *organise*, *organises*, and *organising*

The goal of stemming and lemmatisation is to reduce inflectional forms and transform a word to a common base form.

Stemming: chops off the ends of words in hopes to achieve this common base form.

e.g: car, cars, car's, cars' -> car

Lemmatisation: uses a vocabulary and morphological analysis of words and returns the base or dictionary form of a word (knows as the lemma).

e.g: saw, seeing -> see

To learn more, check out **Porter’s algorithm** for stemming and **WordNet** for lemmatisation!

Feature representation

From text, to tokens, to a representation

e.g. Berkeley Restaurant project sentences

s1: can you tell me about any good restaurants, cantonese restaurants close by?
s2: mid priced thai food is what i’m looking for
s3: tell me about chez panisse
s4: can you give me a listing of the kinds of food that are available

Feature representation

From text, to tokens, to a representation

e.g. Berkeley Restaurant project sentences

s1: “can” “you” “tell” “me” “about” “any” “good” “restaurants” “cantonese” “restaurants” “close” “by”
s2: “mid” “priced” “thai” “food” “is” “what” “i’m” “looking” ~~“for”~~
s3: “tell” “me” “about” “chez” “panisse"
s4: “can” “you” “give” “me” ~~“a”~~ “listing” ~~“of”~~ ~~“the”~~ “kinds” ~~“of”~~ “food” “that” “are” “available”

Feature representation

From text, to tokens, to a representation

e.g. Berkeley Restaurant project sentences

tokens	cantonese	restaurant	thai	food	me	good	…
s1	1	2	0	0	1	1	…
s2	0	0	1	1	0	0	…
s3	0	0	0	0	1	0	…
s4	0	0	0	1	1	0	…

Feature representation

This representation is called a **bag of words** model.

Set of tokens is called a **dictionary**.

Vectorising a sentence means expressing the sentence in terms of the dictionary, either as a binary entry or a real value in [0,1].

Feature representation

Instead of a raw count of word appearance, we can have a better way to represent the words.

Term frequency-inverse document frequency is a numerical statistic that intends to reflect how important a word is to a document in a collection of documents.

Feature representation

Term frequency: f(tf(f,d))

tf(f,d)= $$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Inverse document frequency: g(idf(t,D))

idf(t,D) = $$\log \frac{N}{|\{d \in D : t \in d\}|}$$

TF-IDF: f(tf(f,d)) X g(idf(t,D))

f(.), g(.) are some transformation of the measures, **e.g:** smoothing, scaling, etc…

Feature representation

- Two big limitations of this representation:
  - sequential nature is lost

    e.g: “is this true” - “this is true” have the exact same representation

  - potential scalability issues
- Can improve with using N-grams instead of 1-gram

  e.g: 2-gram

  “is this true” -> “is this” “this true”

  “this is true” -> “this is” “is true”

Text classification

This is a supervised Machine Learning problem.

Input:

- document **d**
- fixed set of classes **C = {c1, c2,..., cJ}**
- A training set of **m** hand-labeled documents **(d1,c1),....,(dm,cm)**

Output:

- a learner classifier $\gamma : d \to c$

Text classification

- Classifier: **Naive bayes**
  - Simple classification method based on Bayes rule
  - Relies on very simple representation of document: Bag of words
  - For a document **d** and a class **c:**

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

Text classification - Probabilistic models

$$
\begin{aligned}
c_{MAP} &= \arg\max_{c \in C} P(c|d) \\
&= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} \\
&\propto \arg\max_{c \in C} P(d|c)P(c)
\end{aligned}
$$

Text classification - Probabilistic models

$$
\begin{aligned}
c_{MAP} &\propto \arg\max_{c \in C} P(d|c)P(c) \\
&= \arg\max_{c \in C} P(x_1, x_2, ..., x_n|c)P(c)
\end{aligned}
$$

Text classification - Probabilistic models

$$c_{MAP} \propto \arg\max_{c \in C} P(d|c)P(c)$$
$$= \arg\max_{c \in C} P(x_1, x_2, ..., x_n|c)P(c)$$

Assuming conditional independence

$$P(x_1, x_2, ..., x_n|c) = P(x_1|c) \cdot P(x_2|c) \cdot ... \cdot P(x_n|c)$$

$$c_{NB} = \arg\max_{c \in C} P(c) \prod_{x \in X} P(x|c)$$

Text classification - Probabilistic models

$$c_{NB} = \arg\max_{c \in C} P(c) \prod_{x \in X} P(x|c)$$

We need to estimate the values of P(c) and P(x|c)

Maximum likelihood estimates:

$$\hat{P}(c_j) = \frac{|\{\text{d s.t. label(d)} = c_j\}|}{N}$$

proportion of documents c_j

$$\hat{P}(x_i|c_j) = \frac{count(x_i, c_j)}{\sum_{x \in V} count(w, c_j)}$$

fraction of times word x_i appears among all words in documents of topic c_j

Text classification - Probabilistic models

$$\hat{P}(x_i|c_j) = \frac{count(x_i, c_j)}{\sum_{x \in V} count(w, c_j)}$$

fraction of times word x_i appears among all words in documents of topic c_j

What if x_i never appears in a document of class c_j ?

$$\hat{P}(x_i|c_j) = \frac{count(x_i, c_j) + 1}{(\sum_{x \in V} count(x, c_j) + 1)}$$

Laplace smoothing

Naive Bayes

- Naïve bayes classifiers can use any sort of feature:

  e.g: URL, email address, dictionaries, network features…

- But only word features

- Naïve bayes has an important similarity to **language modelling**.

Naive Bayes

- Very Fast, low storage requirements
- Robust to Irrelevant Features
  - Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features
  - Decision Trees suffer from fragmentation in such cases – especially if not much data
- In general, a good dependable baseline for text classification

Other classifiers

- SVM
- Random Forest
- Adaboost

Text classification - Pipeline

1. From training corpus, extract **dictionary**.
2. Create **vectoriser** to transform documents into vectors
3. Using Training data {(d,l)}, train a **classifier** of choice
4. Given a test set {d'}:
    1. vectorise documents using vectorizer
    2. use classifier to predict class:
       i.e. find the label c which maximises probability P(c|d)

Document Similarity

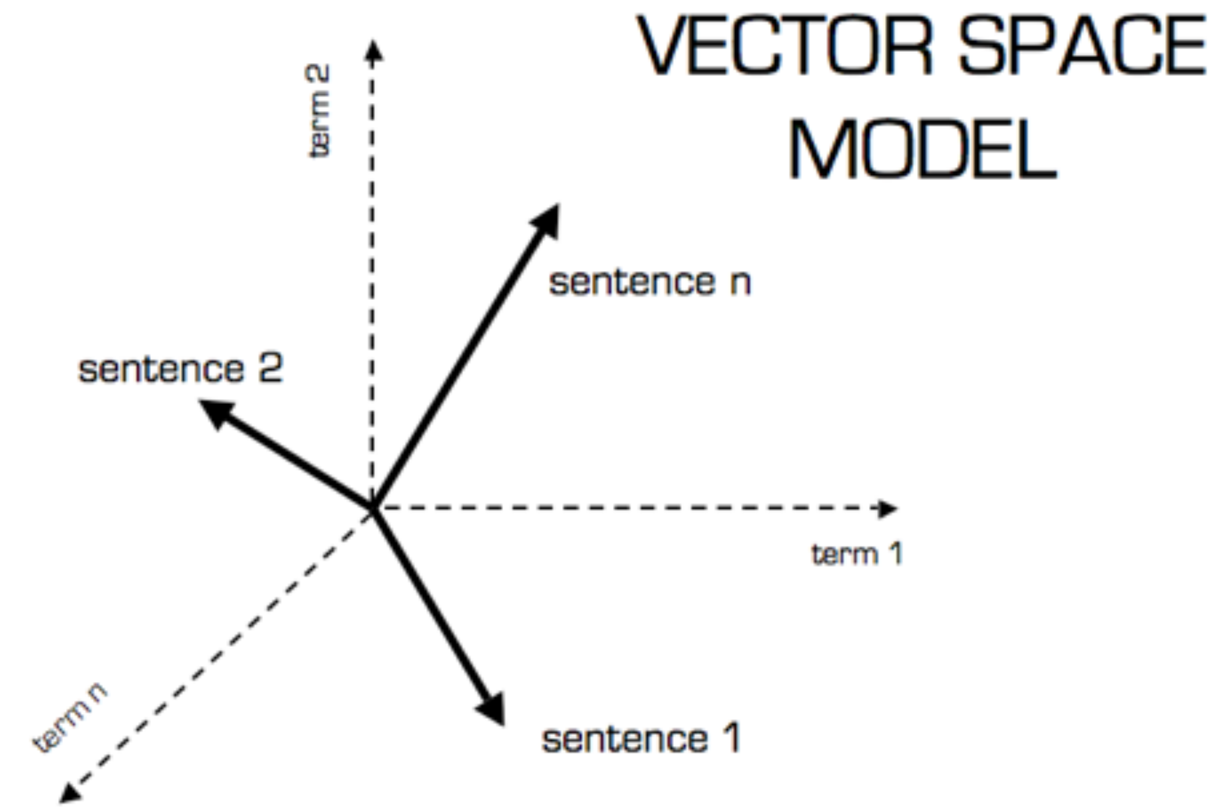
- We want to find documents which are similar to each other (common in search) but we don’t necessarily have labelled data

e.g:

- Information retrieval problem:
  - Issue search query **Q**
  - Find set of documents {d} which are relevant to query **Q**

Document Similarity

- Suppose we have a vector representation of **Q** and documents {**d**}.
- These live in a vector space.
- We can compute the distance between two vectors using the cosine distance



$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Performance metrics

- accuracy:

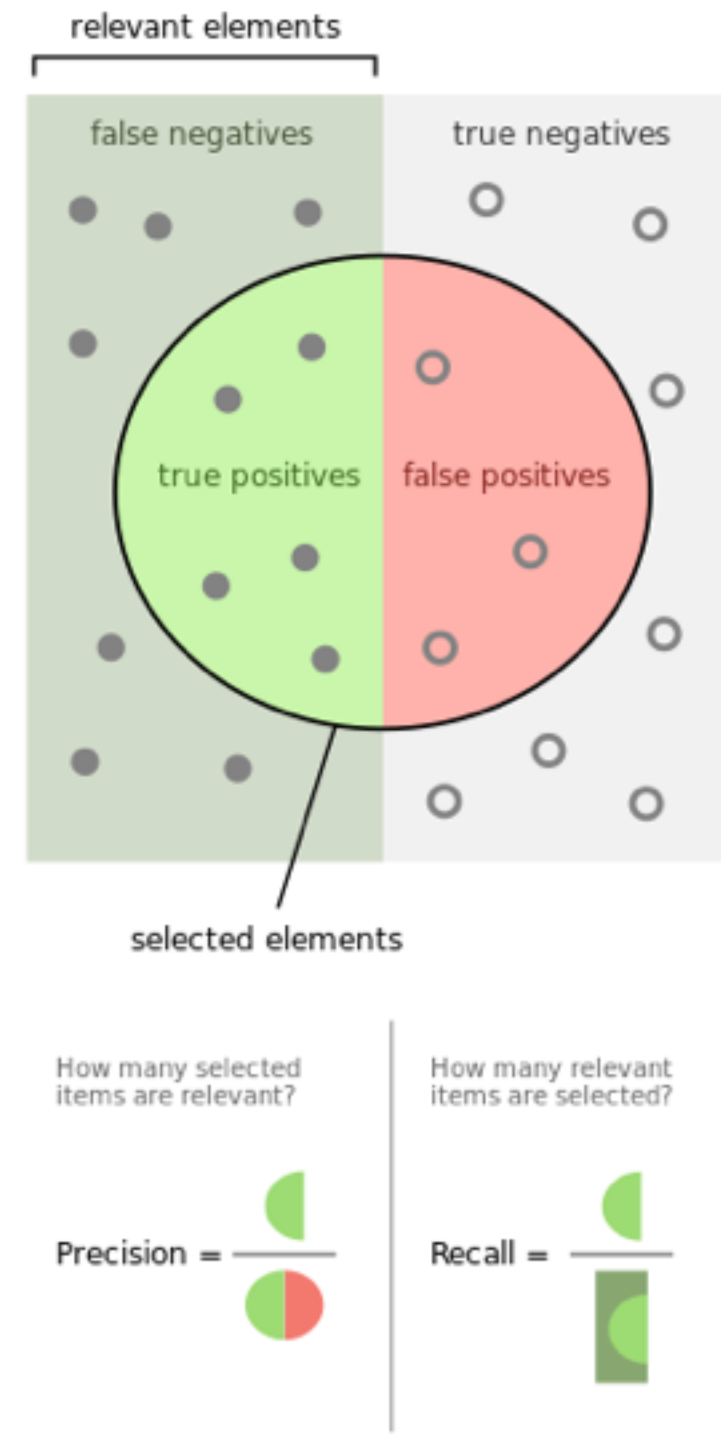
$$\frac{TP + TN}{total}$$

- precision:

$$\frac{TP}{TP + FP}$$

- recall:

$$\frac{TP}{TP + FN}$$



PART 2: PRACTICE

Exercise: Text classification

	Doc	Words	Class
Training	1	mathematics stochastic mathematics	m
	2	mathematics mathematics algebra	m
	3	algebra groups	m
	4	schrodinger quantum algebra	p
Test	5	mathematics mathematics mathematics quantum schrodinger	?

$$\hat{P}(c) = \frac{N_c}{N}$$

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

What's the likely class for document 5?

Exercise: document classification & document clustering

git clone https://github.com/hanveiga/nlp-class-2017.git