Linguistique computationnelle

Classifying Documents

Based on Dickinson, Brew, & Meurers (2013)

Avril 2021

Document classification

Document classification = sort documents into user-defined **classes**

- e.g., email sent to the *New York Times* could be classified into letters to the editor, new subscription requests, complaints about undelivered papers, job inquiries, proposals to buy ad pages, and others

Consider the case of **sentiment analysis**:

- automate the detection of positive and negative statements in documents
- would allow one to track opinions about policies, products, & positions

Sentiment Analysis

Example #1

For the movie *Pearl Harbor*:

> *Ridiculous movie. Worst movie I've seen in my entire life [Koen D. on metacritic]*

Sentiment Analysis

Example #2

One of my favorite movies. It's a bit on the lengthy side, sure. But its made up of a really great cast which, for me, just brings it all together. [Erica H., again on metacritic]

Sentiment Analysis

Example #3

The Japanese sneak attack on Pearl Harbor that brought the United States into World War II has inspired a splendid movie, full of vivid performances and unforgettable scenes, a movie that uses the coming of war as a backdrop for individual stories of love, ambition, heroism and betrayal. The name of that movie is "From Here to Eternity." (First lines of Alan Scott's review of "Pearl Harbor", New York Times, May 25, 2001)

Sentiment Analysis

Example #4

The film is not as painful as a blow to the head, but it will cost you up to $10, and it takes three hours. The first hour and forty-five minutes establishes one of the most banal love triangles ever put to film. Childhood friends Rafe McCawley and Danny Walker (Ben Affleck and Josh Hartnett) both find themselves in love with the same woman, Evelyn Johnson (Kate Beckinsale). [Heather Feher, from www.filmstew.com]

Some document classification tasks

- **Sentiment analysis**: what is the attitude of the text?
- **Authorship attribution**: who wrote a text?
  - Author Identification (who penned *The Federalist Papers*?)
  - Forensic Evidence (who wrote the note?)
  - Plagiarism Detection (who did the work?)
- **Spam filtering**: is this email junk or not?
- **Language identification**: which language is this document in?

Language identification

Let's consider this relatively simple task first . . .

- One can sometimes tell the language used by
  - which characters are used,
    - e.g. *Liebe Grüße* uses ü and ß → German
  - which character encoding is being used
    - e.g., ISO 8859-8 is used to encode Hebrew characters → text is written in Hebrew
- But how can you tell if you are reading English vs. Japanese transliterated into the Roman alphabet? Or Swedish vs. Norwegian?

Language identification

N-grams

- One simple technique for identifying languages is to use **n-grams** = stretch of *n* tokens (i.e., letters or words):
  - Go through texts for which we know which language they are written in and store the n-grams of letters found, for a certain *n*.
    - e.g., extracting the trigrams (3-grams) for the last sentence we'd get: *Go* , *o t*, *th*, *thr*, *hro*, *rou*, . . .
  - This provides us with an indication of what sequences of letters are possible in a given language (and how frequent they occur).
    - e.g., *thr* is not a likely Japanese string.
- How do we make this more concrete?

Language identification

Frequency distributions

- Store a **frequency distribution** of trigrams, i.e., how many times each n-gram appears for a given language.

n-gram	English	Japanese
aba	12	54
ace	95	10
act	45	1
arc	8	0
...	...	

- Now, apply the frequency distribution to a new text and use it to help calculate the probability of the text being a particular language.
  - Compare each n-gram to see if it is more likely to be English or Japanese.
  - See which language won the most comparisons.

Machine Learning

Document classification is an example of a computer science activity called **machine learning**, which is itself part of the subfield of **artificial intelligence**

- We have access to a **training set** of examples, from which we will learn
  - e.g., articles from the on-line version of last month's New York Times
- Long-term goal: use what we have learned to build a robust system that can process future examples of the same kind
  - e.g., articles that are going to appear in next month's New York Times
- As an approximation, we use a separate **test set** of examples to stand in for the unavailable future ones
  - e.g., this month's New York Times articles
  - Since the test set is separate from the training set, the system will not have seen them.

Supervised Learning

Supervised learning: training set and test set have been labeled with desired "correct answers"

- News service provides a stream of uncategorised articles, & we want to sort them into "News", "Sports", "Arts", "Business" and "Do not use"

1. Label a few hundred articles with the desired categories, to make a training set and a test set
2. Apply machine learning software to the labeled training set.
   - This produces an object called a **model**: summarizes what has been learned from training set
3. Read learned model into machine learning software and use it to generate **predictions** for test set
4. Deploy the learned model on unseen examples
   - Model uses what it has learned to sort the articles into the necessary piles

Unsupervised Learning

Unsupervised learning: assume there are no pre-specified categories.

- Newspaper still gets a stream of uncategorised articles, but ...
- Task now is to organize the articles into piles in such a way that **similar** articles occur in the same pile.
  - Piles are often called **clusters**, and the process of organizing articles into clusters is called **clustering**
- Clusters might share some property, e.g., being about sports
  - Algorithm just groups articles; it does not name them

Unsupervised Learning (cont.)

Advantage of unsupervised learning:

- You do not need a training set, so there is no costly process of going through labeling up the articles

Possible disadvantage of unsupervised learning:

- Clusters may not be intuitive
  - e.g., clustering common words often gives *Monday, Tuesday, Wednesday, Thursday* in one cluster, with *Friday* in another
  - Friday is the only weekday that frequently turns up following “Thank goodness it is . . . ”

Clustering is also difficult to evaluate

Features & Evidence

First step in classifying or clustering documents:

- identify properties most relevant to the decision we want to make, i.e., **features**
  - in biology: specimen has features like scales or gills → observing these tells us it's a fish

For spam filtering: could have features such as:

- Whether the document mentions a large sum of money
- Whether the greeting used in the document is something weird like "Respected Madam" or not
- Whether it has words written entirely in upper-case
- Whether the document uses the words "Viagra" and "sex" close to each other

None of these features are certain indicators of spam

- but all provide evidence that the document is more likely to be spam than if we had not seen the feature

Features

To make a useful system we need to tell the computer two things:

1. Exactly which features are used and exactly how to detect them
   - **feature engineering**
   - Hard to automate this step
2. How to weight the evidence provided by the features
   - Often works well to use machine learning for this

Feature engineering

Two common strategies for doing feature engineering:

1. **Kitchen sink** strategy: use lots of features, in the hope that some of them will be relevant and useful
   - e.g., throw every possible absence/presence of a word feature at a spam detector
   - Need to choose a machine learning method that is good at:
     - focusing on the few but important relevant features
     - ignoring the many irrelevant features

Advantage of using words as features in spam detection:

- It's almost as easy to collect and count all the words in a document as it is to collect just a selected few
- Collecting other features may be more complicated

Feature engineering (2)

2. **Hand-crafted** strategy: use careful thought to try to identify a small set of features likely to be relevant
   - advantage: fewer irrelevant features, so machine learning method doesn't have to be as good at ignoring
   - disadvantage: task of choosing features is difficult

Iterative method:

1. Pick initial set of features
2. Train a classifier & measure how well it does
3. Work out which features are working well, and which less well, leading to ideas for further features

Feature engineering (3)

The best features may be hard to collect reliably

- How exactly does one write reliable code for deciding whether a document is offering to improve the reader's sex life?
- Good predictor for spam, but hard to reliably detect

May be better to quickly collect a large number of easy but marginally relevant features than to extract the difficult features

Measuring success: accuracy

Classification task: every item must be attributed one category. There is no limit in the number of categories.

n	Number of items (= number of predictions)
h	(*hits*) Number of correct predictions

$$\text{Accuracy} = \frac{h}{n}$$

Precision/Recall

For a task of “spotting”: among items that are given I am in charge of finding (all and only) items of a specific kind.

N	Total number of items
n	Number of items to be found $n \leqslant N$
h	(*hits*) Number of correct identifications (true positives) $h \leqslant n$
f	(*false alarms*) Number of false identifications (false positives)
m	(*misses*) Number of items not found (false negatives) $m + h = n$
c	total number of decisions made ($c = h + f$)

Precision

$$p = \frac{h}{h+f} = \frac{h}{c}$$

Recall

$$r = \frac{h}{m+h} = \frac{h}{n}$$

f-score: harmonic mean:

$$f = 2 \times \frac{p.r}{p+r}$$

Measuring sucess

Running a classifier on a document to decide whether it is spam or not is similar to medical diagnostic tests

Here there are two two-way distinctions to be made:

1. The test can come out either positive or negative.
2. The patient may or may not really have the disease.

	Has disease	No disease
Test positive	True positives	False positives
Test negative	False negatives	True negatives

Measuring sucess (cont.)

	Has disease	No disease	
Test positive	True positives	False positives	**Positive predictive value**
Test negative	False negatives	True negatives	**Negative predictive value**
	Sensitivity	**Specificity**	

Measuring sucess (cont.)

Sensitivity is the same thing as recall:

$$\text{Sensitivity} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

The positive predictive value is the same as precision:

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

Specificity helps deal with not having the disease:

$$\text{Specificity} = \frac{\text{True negatives}}{\text{True negatives} + \text{False positives}}$$

Document classifiers

Naive Bayes

Recall that we need to:

- collect & assess evidence about the appropriate classification for documents
- i.e., need to choose an appropriate algorithm for weighting the evidence

Naïve Bayes for document classification:

- A competition between the hypothesis that the document is a piece of junk mail & alternative hypothesis that it is not.
- Expressed by doing a probability calculation for each of the two hypotheses
  - based on the evidence we collect

Example data

Pretend we have collected statistics for just a few words

- Imaginary user Sandy who chats about horses

	Spam	Ham
cash	200	3
Alice	1	50
Seth	2	34
Emily	2	25
Viagra	20	0
credit	12	2
unicorn	0	5
cookie	1	5
hippogriff	0	18
pony	9	50
stallion	3	8
TOTAL	250	200

Bag of words assumption

How does a classifier reconcile conflicting evidence?

Simplest policy: pretend that we are dealing with a completely unstructured collection of words

- **bag of words assumption**
- Ignore the fact that the words were arranged into a particular order, forming sentences and paragraphs

Imagine that we cut up a document & put the words in a bag

- Draw words out of the bag & ask whether it is more likely to have come from a spam document or not

Example calculation

Imagine that word from the bag was “Emily”

- Seen the word in spam 2 times, out of 250 total spam words
- Likely to see “Emily” 2 times in 250 (0.8%) if the document that we put in the bag was spam.

Non-spam:

- Seen the word 25 times in Sandy’s real messages, out of 200 total non-spam words
- Likely to see the word 25 times in 200 (12.5%) if the document is not spam

12.5% is much bigger than 0.8%: from seeing just one word, the document in the bag is more likely to be ham

- Record the **odds ratio** for ham to spam as 12.5/0.8, or nearly 16 (much greater than 1)

Example calculation (cont.)

Suppose that the next word is “credit”

- 12 in 250 for spam
- 2 in 200 for ham
- Odds ratio for this word: 2/200 against 12/250, or about 0.29
  - Less than 1, so we think, on the basis of this word alone, that the document in the bag is probably spam

To combine the evidence, we multiply the ratios

- $16 \times 0.29 = 4.63$
- Combined ratio is greater than 1: the two words together indicate a genuine document and not spam

Continue to calculate a ratio for each new word as it comes out of the bag, & multiplying it into the combined ratio

Document classifiers

Perceptron

Idea of Naïve Bayes: count things that occur in the test set

Different idea: *error-driven learning*, specifically the **perceptron**:

- Maintains a collection of **weights**
- Each weight links a feature with an **outcome**
  - Perceptron learns from experience: predict outcomes & then adjust the weights when it makes a wrong prediction

Initially: weights are uninformative

- Over time the perceptron builds up an ability to associate features with outcomes in a useful way

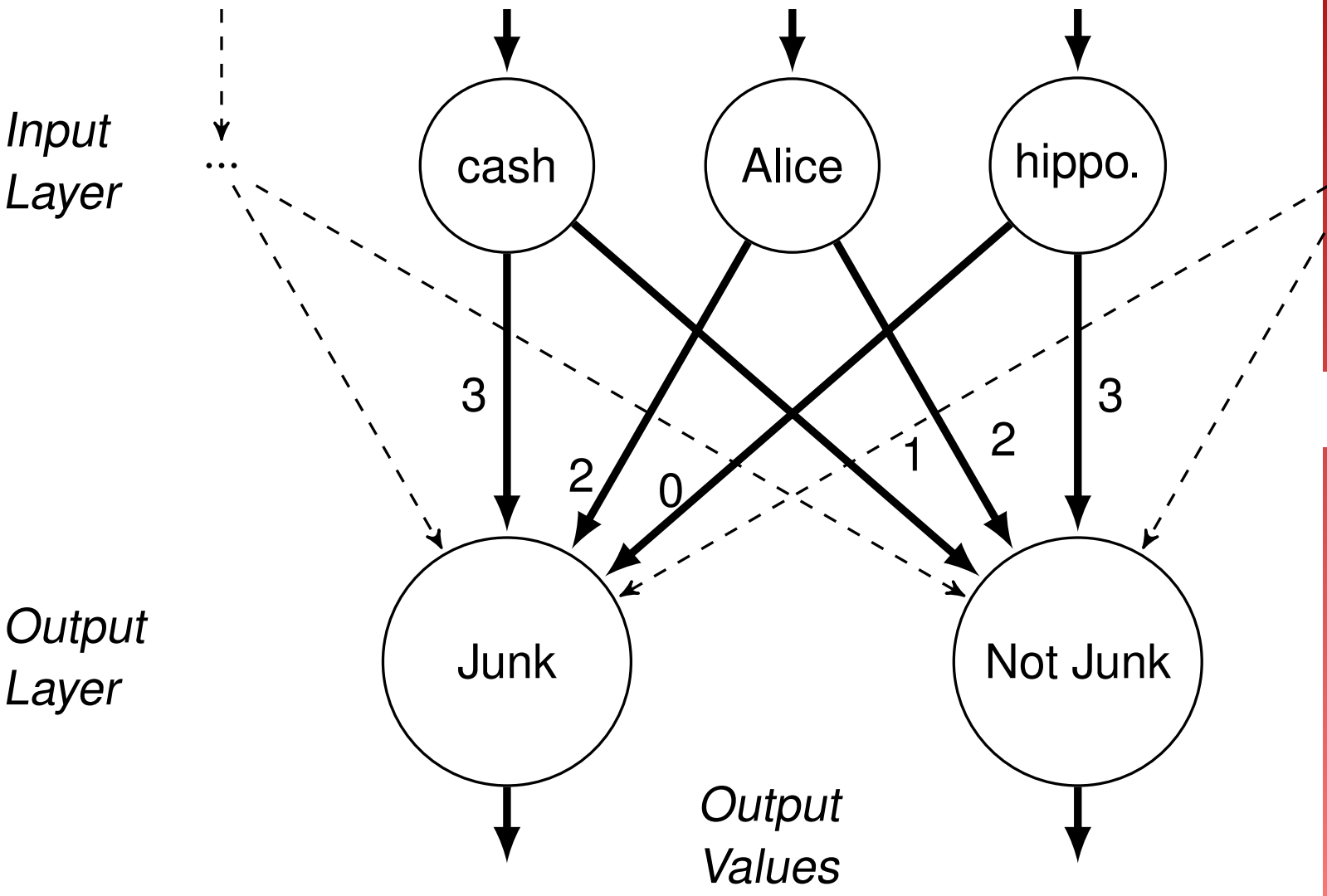
Perceptron: layered network

Perceptron: network with two layers

- **Input layer** has one node for each possible input feature
  - e.g., one node for each of “cash”, “Alice”, “Seth”, “hippogriff”, and so on
- **Output layer** contains one node for each possible outcome
  - e.g., one each for “Junk” and “Not junk”

Edges linking input & output layers are associated with weights (not shown in the diagram)

Perceptron



Perceptron weights

In order to predict, the perceptron reads a document, and notes which words are present

- We turn the nodes that correspond to these words on, and turn the others off
- The weights decide how strongly to transmit the activity of the active nodes to the output layer

Word	Junk	Not Junk
cash	3	1
Alice	2	2
hippogriff	0	3

- total activity of "Junk" output node: $3 + 2 + 0 = 5$
- total activity of "Not Junk" output node: $1 + 2 + 3 = 6$

If "Not Junk" is right prediction, perceptron stays as it is; if message is actually junk, the weights need to change

Perceptron weight updating

Let us suppose perceptron is wrong

- Perceptron algorithm changes all the relevant weights a little bit, moving the result closer to a correct prediction
  - Increase the weight of “cash” (and each of the other two words) as a predictor for “Junk”
  - downweight it as a predictor for “Not Junk”

Word	Junk	Not Junk
cash	3.01	0.99
Alice	2.01	1.99
hippogriff	0.01	2.99

Perceptron weight updating (cont.)

To train the perceptron, we go through the training corpus

- Present each example to current version of perceptron
- Adapt weights whenever we make a mistake
- When we get to the end of the training corpus, we start again at the beginning
  - Each round of this process is called an **iteration**

After a sufficient number of iterations, weights change enough to flip some predictions & mistakes tend to go away

Authorship Attribution

Switching gears to **authorship attribution** . . .

- In a classic study, Mosteller and Wallace (1964) applied authorship detection techniques to *The Federalist Papers*.
- *The Federalist Papers* were a series of 85 articles written between 1787 and 1788 by James Madison, Alexander Hamilton and John Jay to persuade New York to ratify the Constitution.
- Some of the papers were clearly written by one of the three; 12 are in question, written either by Hamilton or Madison.
- Mosteller and Wallace examined the frequency of various words in the disputed papers and compared each to a model of known Hamilton writings and known Madison writings.

Stylometry

- **Stylometry** defines the features of an author's style and measures those features in two or more texts to determine the similarity between the texts.
- The more similar the styles, the more likely two texts are to be written by the same author.
- The idea is that style operates at a subconscious level, which makes it more consistent (and perhaps measurable?).
- In other words, writing style is a "linguistic fingerprint."

Stylometric Approach

- The basic approach:
  - Extract style markers
  - Use the markers to classify texts
- Style markers may be based on words, grammar or a combination.

Lexical Style Markers

- **Lexical style markers** are words that give clues about authorship.
- There are two types of markers: vocabulary richness and frequency of function words.
  - **Function words** such as “to” and “that” carry little meaning but occur often in a document
  - Function words are independent of topic, but the idea is that *which* function words you choose and *where* you use them are enough to identify you as an author.
- How can we use lexical markers to detect plagiarism?

Frequency of Function Words

- An example of two authors' use of function words, gathered from AP news stories by Zhao and Zobel (2005).

	a	and	for	in	is	of	that	the
Barry Schweid	6.28	9.22	4.94	6.50	1.62	14.66	1.89	29.13
Don Kendall	9.75	7.08	2.36	7.99	3.05	13.16	5.73	41.29

- The Signature Text Analysis program (http://www.philocomp.net/humanities/signature.htm) is designed to help you determine such stylometric indicators

What is plagiarism?

- Clough (2003) defines **text reuse** is the deliberate *or* unintentional use of existing text for the creation of a new text.
  - **Plagiarism** is one kind of text reuse.
  - Reusing newswire text in journalistic publications is another instance of text reuse.

Types of Plagiarism

Clough (2003) outlines six forms of plagiarism:

1. Word-for-word – Whole phrases, sentences or passages are copied, but not attributed.
2. Paraphrasing – The unattributed source material is rewritten, but is still recognizable in the new text.
3. Secondary Source – Sources are cited, but extracted from a secondary source (not the original).
4. Source Form – A source's argument structure/text organization is copied.
5. Ideas – Thoughts (independent of form) are copied without attribution.
6. Authorship – Authorship of an entire text is falsely claimed.

Word-for-Word Plagiarism: Source*

The Passage as It Appears in the Source:

Critical care nurses function in a hierarchy of roles. In this open heart surgery unit, the nurse manager hires and fires the nursing personnel. The nurse manager does not directly care for patients but follows the progress of unusual or long-term patients. On each shift a nurse assumes the role of resource nurse. This person oversees the hour-by-hour functioning of the unit as a whole, such as considering expected admissions and discharges of patients, ascertaining that beds are available for patients in the operating room, and covering sick calls. . . . (Chase, 1995, p. 156)

*Example From the Writing Center at University of Wisconsin-Madison (http://www.wisc.edu/writing/Handbook/QPA_paraphrase.html).

Word-for-Word Plagiarism: Copy

Critical care nurses have a hierarchy of roles. The nurse manager hires and fires nurses. S/he does not directly care for patients but does follow unusual or long-term cases. On each shift a resource nurse attends to the functioning of the unit as a whole, such as making sure beds are available in the operating room, and also has a patient assignment. . . .

Recognizing Plagiarism (1)

The following factors may indicate plagiarism:

- Vocabulary use beyond the skill level of the writer (Ex: technical/advanced terms).
- A drastic change in the quality of writing compared to previous submissions.
- Style or vocabulary inconsistencies within a text.
- Choppy text that lacks transitions or smooth flow, indicating a “cut-and-paste” job.

Recognizing Plagiarism (2)

- Significant similarity between multiple submissions.
- Similar errors between multiple submissions (Ex: the same spelling/grammar errors).
- References that appear in the text but not the bibliography.
- Lack of a consistent bibliographic style within the body or references section of text.

Plagiarism Detection

1. Detection in a single text:
   - Identify inconsistencies within a text
   - Find sources for the inconsistencies
2. Detection across multiple texts:
   - Identify unacceptable collaborations
   - Identify direct copying

Indicators of plagiarism

- The idea is that the more similar two texts are, the more likely it is that one of the text is derived (plagiarized) from the other.
- Possible indicators include vocabulary use, word length, syllable structure, rhyme and grammar
  - Q: what features or methods would you use to detect the similarities/differences between 2 texts?
- Indicators are used to flag texts for later human inspection.

References

The authorship slides were developed (by Stacey Bailey) using the following sources:

- Clough, Paul. 2003. *Old and new challenges in automatic plagiarism detection.* National Plagiarism Advisory Service.
- Keselj, Vlado, Peng, Fuchun, Cercone, Nick and Thomas, Calvin. 2003. N-gram-based author profiles for authorship attribution. In *Proceeding of the Pacific Association for Computational Linguistics* (PACLING'03), Dalhousie University, Halifax, Nova Scotia, Canada, August 2003.
- Putnins, Talis, Signoriello, Domenic, Jain, Samant, Berryman, Matthew and Abbott, Derek. 2005. Advanced text authorship detection methods and their application to biblical texts. In *Proc. SPIE: Complex Systems 6039*. Brisbane, Qld., Australia, December 11-14, 2005.
- Zhao, Ying and Zobel, Justin. 2005. Effective and scalable authorship attribution using function words. In *Proceedings of the AIRS Asia Information Retrieval Symposium*, Jeju Island, Korea, October 2005. pp. 174-189.