JM2050 – Natural Language Processing

Text representation

NLP tasks I

September 19^{th}, 2024

Prof.dr.ir. U. Kaymak

Jheronimus Academy of Data Science

Recap

- Text normalization
  - Removing numbers, punctuation and special characters
  - Convert into lower case
  - Lemmatization and stemming
- Pre-processing text
  - Document filtering
  - OCR and text cleaning
  - Document structuring
  - Tokenization
  - Stop-word removal
- Regular expressions are a powerful method for pre-processing text

[2]

www.jads.nl

Jheronimus Academy of Data Science

Outline

- Representing text
  - Bag of words
  - N-grams
  - Word embeddings
- Examples of NLP tasks
  - Part-of-Speech (POS) tagging
  - Classification

www.jads.nl

Representing text

[4]

With contributions from N. Moonen, D. Kapitan, E. Tromp

Jheronimus Academy of Data Science

Vector space model (VSM)

A fundamental problem of text mining is how to represent the text documents to make them mathematically computable

- Vector Space Model (VSM) aims representing each text document by a numerical vector
- Similarity between vectors quantify the similarity between documents

"cat"

[0.4 -0.23 0.5 1.23 2.8 0.68 0.4]

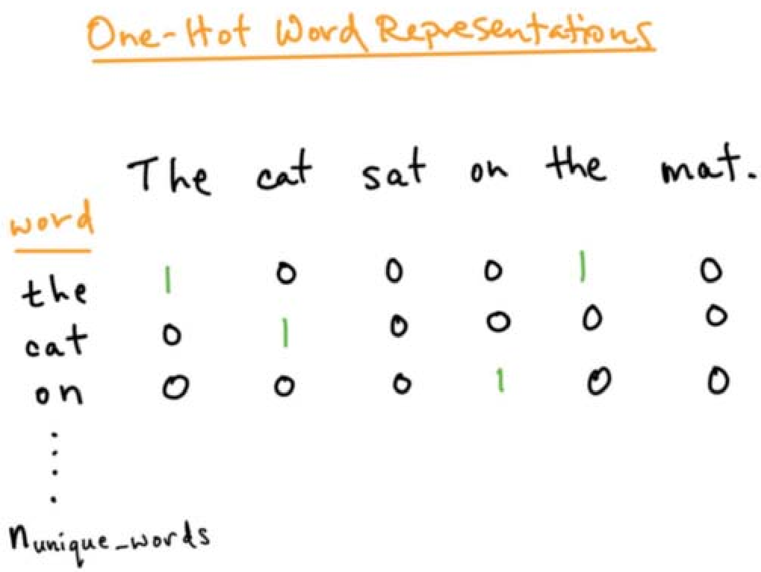
[5]

Yan, J. (2009). Text Representation. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_420

One-hot encoding

- Vector representation of words in a vocabulary
- Each word is represented by a vector of size n (number of words in the vocabulary)

- Memory inefficient
- Sparsity
- Context of a word is not considered



https://www.enjoyalgorithms.com/blog/word-vector-encoding-in-nlp

[6]

www.jads.nl

Bag-of-Words (BOW)

Discover useful information from text data, such as patterns in the texts.

Simplify text by counting words
- Ignore word order

+ Simple, statistical properties, scalable to large corpora
- Sparsity, poor semantics

www.jads.nl

Document-Term Matrix (DTM) fitted with a count-based measure

$$\begin{bmatrix} & T_1 & T_2 & \cdots & T_t \\ D_1 & w_{11} & w_{21} & \cdots & w_{t1} \\ D_2 & w_{12} & w_{22} & \cdots & w_{t2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D_n & w_{1n} & w_{2n} & \cdots & w_{tn} \end{bmatrix}$$

DTM is an implementation of BOW

Three common metrics:

- 0 or 1
- Term frequency
- TF-IDF (Term Frequency – Inverse Document Frequency)

DTMs are useful for many machine learning techniques (Regression, DT, RF, SVM, Naïve Bayes, k-NN, Clustering, Topic modeling, NN)

[26]

www.jads.nl

Jheronimus Academy of Data Science

TF-IDF

Multiply these two terms

Term Frequency * Inverse Document Frequency

$$tf(t,d) = f_{t,d} \;\; or \;\; tf(t,d) = \frac{tf(t,d)}{|d|} \qquad idf(t,D) = \log \frac{N}{|\{d \in D : t \in D\}|}$$

Intuition: important words occur a lot in some documents,
but don't occur in many documents (i.e. frequent in document & rare in corpus)

Example: a document with 100 words contains the term fraud 5 times.
The term fraud appears in 200 docs of our corpus of 10 000 docs.

TF: 5/100=0.05, IDF: log (10 000 / (200)) = 1.69
TFIDF: 0.05 * 1.69 = 0.0845

www.jads.nl

BOW – Supervised

Both input and output are known

Algorithm learns mapping between input and output

\+ A priori specifications, marginal effects
\- Pre-existing knowledge, overfitting

Disciplines: computer science, political science

Examples: SVM, Naïve Bayes, logistic regression

- Language detection
- Spam filtering
- Document classification
- Author identification

BOW – Unsupervised

Algorithm automatically creates clusters of similar data, but outcome is unknown

+ Scalable, no pre-existing knowledge of corpus
- Parameter tuning, interpretation

Disciplines: computer science, political science

Examples: k-means clustering, fuzzy c-means clustering, topic modeling

- Features of apps in Google Play/Apple App Store
- Characteristics of products and services based on website data
- Value propositions based on company profiles of Crunchbase

N-gram representation

n-grams are sequences of n words

Original text	To Sherlock Holmes she is always the woman. I have seldom heard him mention her under any other name.
Unigrams (1-grams)	{to, sherlock, holmes, she, is, always, the, woman, I , have …}
Bigrams (2-grams)	{to sherlock, sherlock holmes, holmes she, she is, is always …}
Trigrams (3-grams)	{to sherlock holmes, sherlock holmes she, holmes she is, …}
Quadgrams (4-grams)	{to sherlock holmes she, sherlock holmes she is, holmes she is always …}

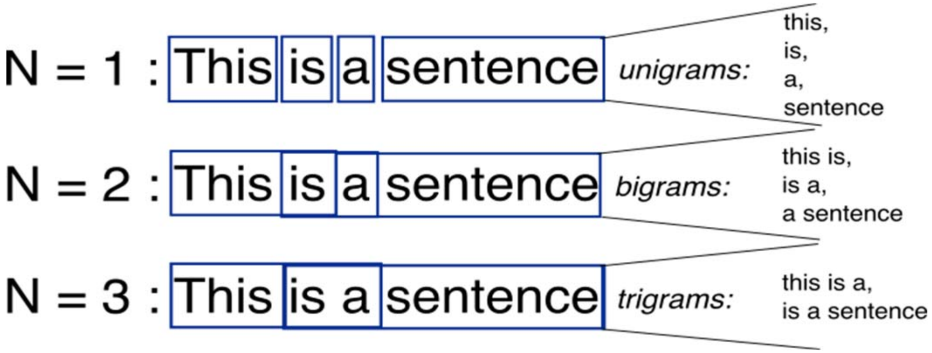
[12]

www.jads.nl

Jheronimus Academy of Data Science

Ngram creation

Useful to identify multi-word concepts



Examples:
Corporate social responsibility
Machine learning
Artificial Intelligence
Business model

Python: ngrams() from nltk package

https://github.com/chris-billingham/topic_modelling/blob/master/gsr_model.R

[41]

www.jads.nl

bag-of-words is basically counting unigrams

unigrams and bag of words

Force of LSTM and GRU – Blog

document term matrix

term	doc_1	doc_2	...
it	6	...	...
I	5	...	...
the	4	...	...
to	3	...	...
and	3	...	...
seen	2	...	...
...	...	...	...

The amazing effectiveness of character n-grams

Language detection

- Character quadgrams can detect language with >99% accuracy

Spam detection

- Character quadgrams and pentagrams can detect spam with up to 95% precision and recall
- Algorithm can be language *independent*

Intuition behind n-grams

Markov assumption

$$P\left(\frac{I\ believe\ coding\ in\ Python\ is\ fun}{I\ believe\ coding\ in\ Python\ is}\right)$$

$$\approx P\left(\frac{Python\ is\ fun}{Python\ is}\right)$$

Using n-grams & TF-IDF in practice

Advantages	Disadvantages
Easy to integrate in pipeline	Document term matrix is very sparse and can get large for n > 1 (using lemmatization helps)
Easy to customize parsing for domain specific texts	No contextual information in available in model
Bi-/tri-/quad grams are surprisingly effective	Can’t deal with out-of-vocabulary words (using lemmatization helps)

17

Word embeddings

How to create vector representations of words without the sparsity of n-grams?

First solution: word2vec

- Developed by four (then) Google engineers in 2013 (original paper on Arxiv)
- Has led to significant breakthroughs in deep learning, e.g. for translations
- Google's pre-trained word2vec model (neural network):
  - 300-dimensional vector space
  - with 10^{11} words
  - vocabulary of 3×10^6 words and phrases

Many variations since then (see overview article or this blogpost)

- GloVe (2014): looks at global word co-occurence
- FastText (2017): also looks at sub-word (character n-grams)

Distributional hypothesis

- In Word2vec, the dimensions of the vector do not have a (clear) meaning

[0.4 -0.23 0.5 1.23 2.8 0.68 0.4]

→ What does this mean?

- **Distributional hypothesis:** words which are synonyms (e.g. *oculist* and *eye-doctor*) tend to occur in the same environment (e.g., near words such as *eye* or *examined*) with the amount of meaning difference between two words ‘corresponding roughly to the amount of difference in their environments’. (Harris, 1954, 1957)

[19]

Word2vec (intuition)

1. Start with random word vectors
2. Try to explain
   a. A word by its surrounding, or
   b. A word's surrounding by that word
3. Update the vectors
4. Repeat from step 1

Continuous Bag of Words

Skip-gram

Word2vec is unsupervised!

[20]

Continuous Bag-of-Words (CBOW)

Explain a word by its surrounding

The

cat

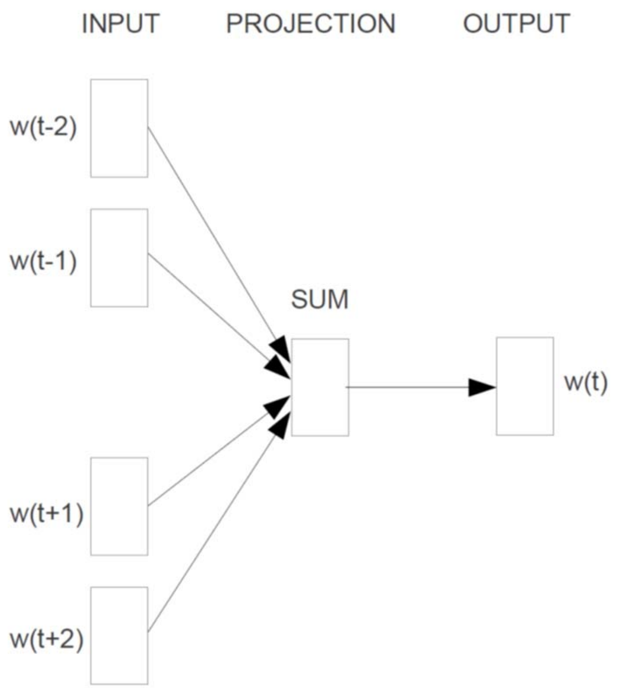
on

the

mat

sat

…

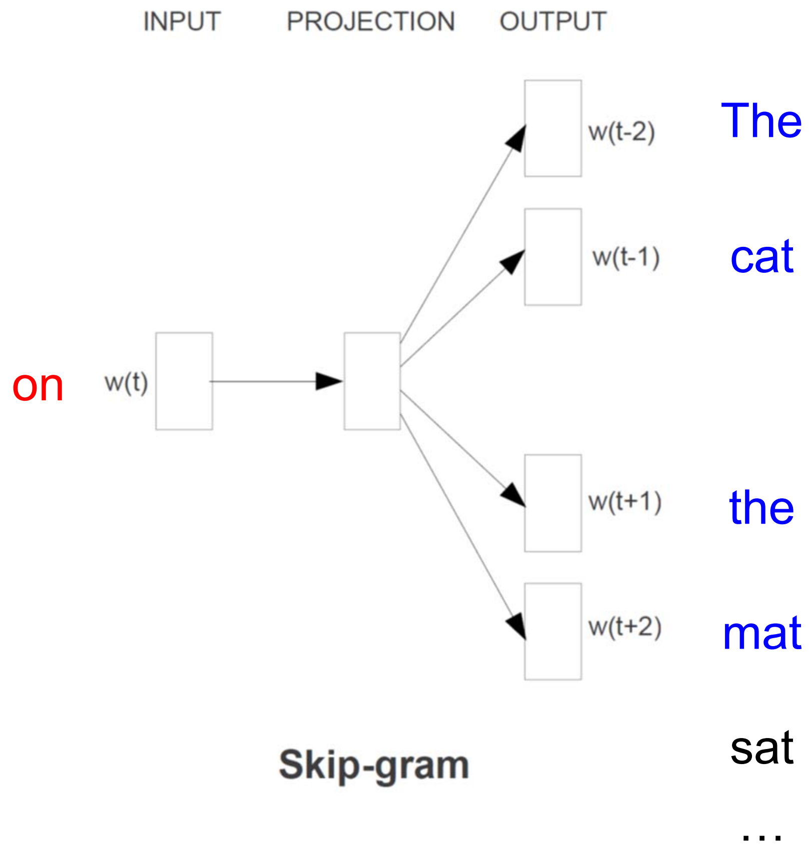


CBOW

[21]

Skip-gram

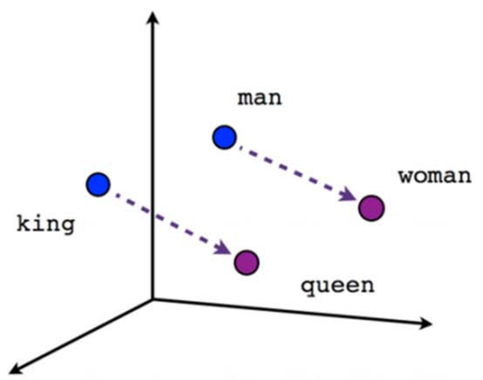
Explain a word's surrounding by that word



[22]

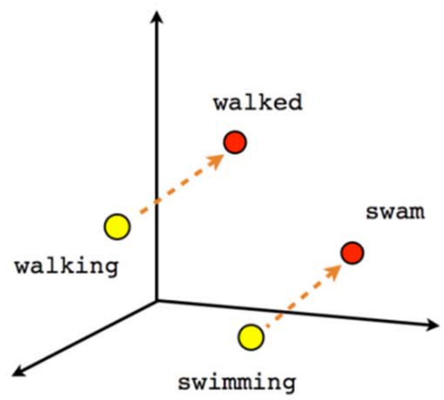
Word2vec result

Semantic word relationships

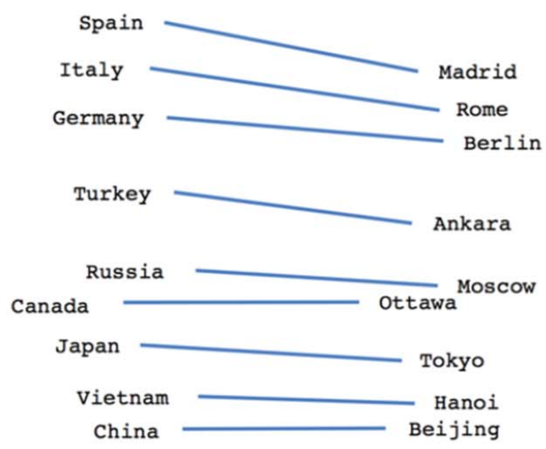


Male-Female

Syntactic word relationships



Verb tense



Country-Capital

23

www.jads.nl

Jheronimus Academy of Data Science

Some Word2vec observations

- Skip-gram is better at semantics
- CBOW is better at syntax
- CBOW is much faster
- Skip-gram is less overfitting to frequent words

[24]

Using word2vec in practice

(More on word embeddings in future lectures)

Advantages	Disadvantages
Many pre-trained models available	Can't deal with multiple meanings of the same word
Easy to integrate in pipeline	Extra care is needed when meanings of words change over time or when using in specific domain
	Can't deal with out-of-vocabulary words

Examples of NLP tasks: Part-of-Speech Tagging

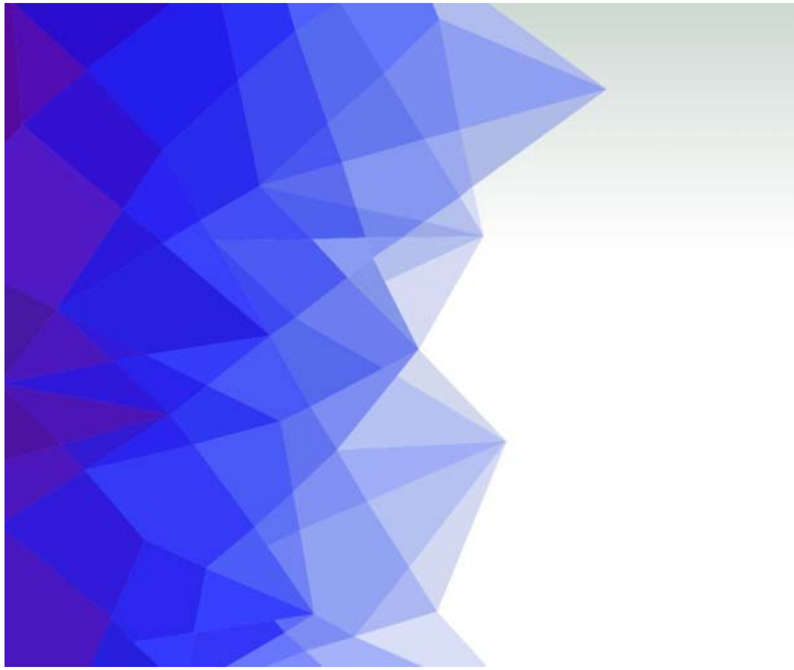
Based on slides from Jurafsky et al. (2020)

26

Jheronimus Academy of Data Science

Parts of Speech

- From the earliest linguistic traditions (Yaska and Panini 5^{th} C. BCE, Aristotle 4^{th} C. BCE), the idea that words can be classified into grammatical categories
- part of speech, word classes, POS, POS tags
- 8 parts of speech attributed to Dionysius Thrax of Alexandria (c. 1^{st} C. BCE)
  - noun, verb, pronoun, preposition, adverb, conjunction, participle, article
- These categories are relevant for NLP
  - Similar contexts of occurrence
  - Can be combined with the same set of morphological suffixes



[27]

www.jads.nl

Jheronimus Academy of Data Science

Two classes of words: Open vs. Closed

Closed class words

- Relatively fixed membership
- Usually **function** words: short, frequent words with grammatical function
  - determiners: ***a, an, the***
  - pronouns: ***she, he, I***
  - prepositions: ***on, under, over, near, by, …***

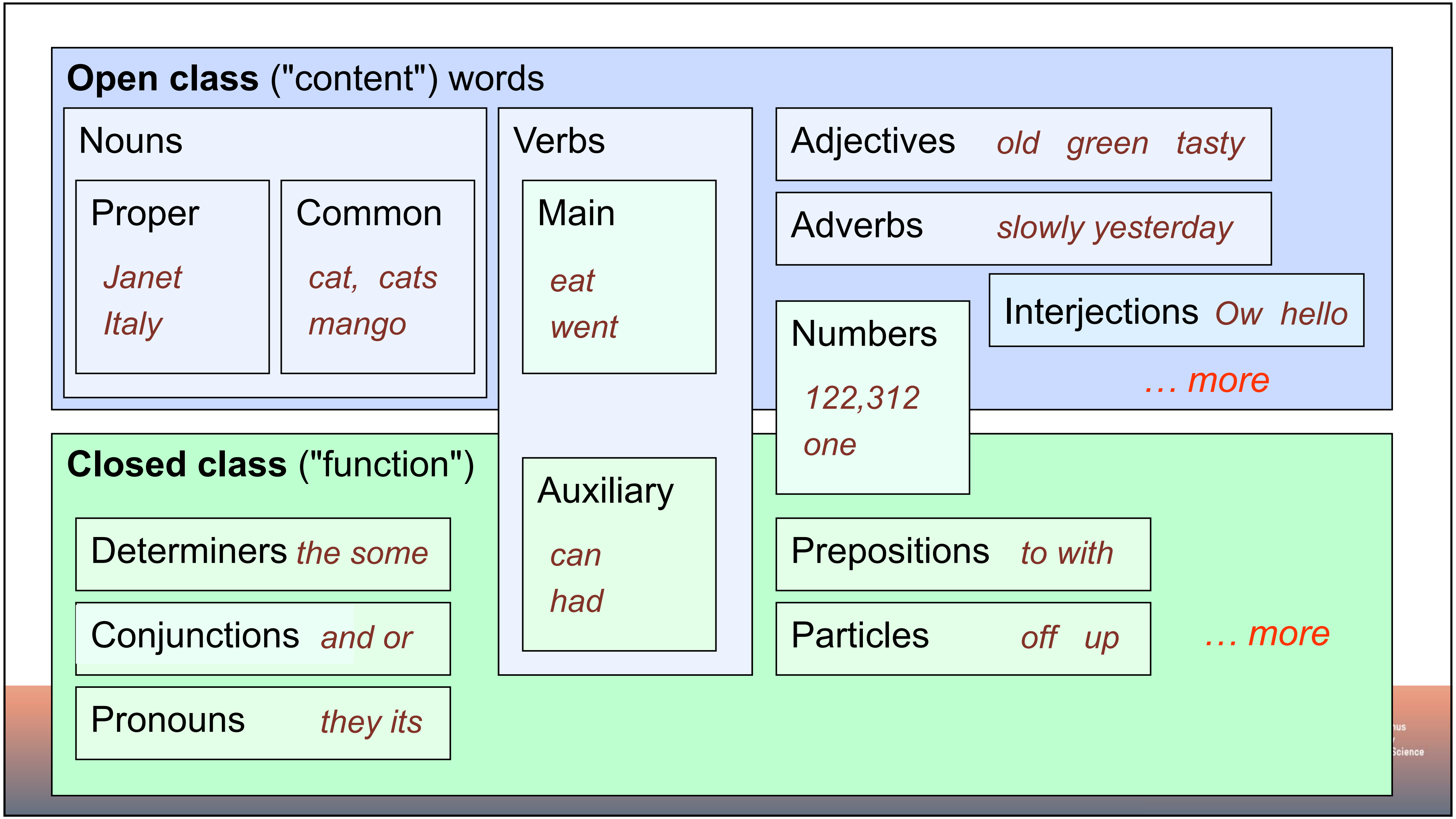
Open class words

- Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
  - Plus interjections: **oh, ouch, uh-huh, yes, hello**
- New nouns and verbs like *iPhone* or *to fax*

[28]

www.jads.nl

Jheronimus Academy of Data Science


Open class ("content") words
Nouns
Proper
Janet
Italy
Common
cat, cats
mango
Verbs
Main
eat
went
Auxiliary
can
had
Adjectives old green tasty
Adverbs slowly yesterday
Interjections Ow hello
Numbers
122,312
one
… more
Closed class ("function")
Determiners the some
Conjunctions and or
Pronouns they its
Prepositions to with
Particles off up
… more

Part-of-Speech Tagging

- Process of automatically assigning a POS tags to words in text
- POS tagger takes a tokenized corpus as input and outputs a sequence of tags, ***one for each input token***
- Words often have more than one POS tag
- **book**:
  - VERB: (***Book*** *that flight*)
  - NOUN: (*Hand me that* ***book***)

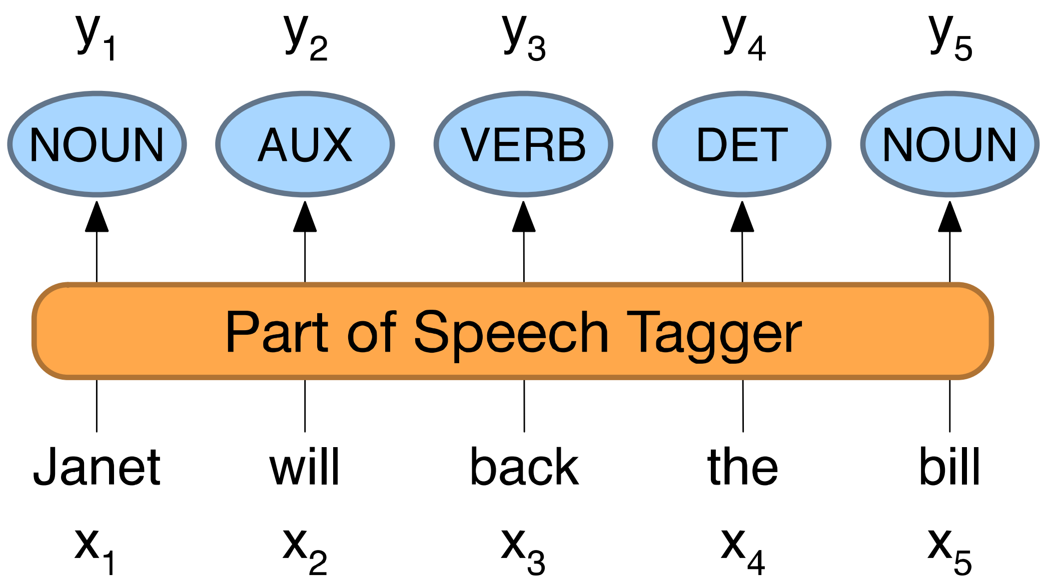
    ➔ disambiguation is needed
- Useful a.o. for lemmatization, word sense disambiguation, Named Entity Recognition, etc.

Why Part of Speech Tagging?

- Can be useful for other NLP tasks
  - Parsing: POS tagging can improve syntactic parsing
  - Machine Translation: reordering of adjectives and nouns (say from Spanish to English)
  - Sentiment or affective tasks: may want to distinguish adjectives or other POS
  - Text-to-speech (how do we pronounce “lead” or "object"?)
- Linguistic or language-analytic computational tasks
  - Need to control for POS when studying linguistic change like creation of new words, or meaning shift
  - Control for POS in measuring meaning similarity or difference

Part-of-Speech Tagging

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



32

"Universal Dependencies" Tagset

Nivre et al. 2016

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

33

Sample "Tagged" English sentences

- There/PRO were/VERB 70/NUM children/NOUN there/ADV ./PUNC
- Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN ’s/PART New/PROPN England/PROPN Journal/PROPN of/ADP Medicine/PROPN

Various annotated corpora exist, especially for English

www.jads.nl

Jheronimus Academy of Data Science

The Penn Treebank tagset

- English-specific, includes 45 tags, typically appended to a word using a slash /
- Doesn't include syntactic information (e.g., is a word the subject or the object?)
- Assumes tokenized input (with multiword expressions tokenized at the white space)

[35]

Other tagged corpora

Some corpora provide information about the PoS of every word in it

- Brown corpus (1million words, balanced)
- WSJ (1million words, news articles)
- Switchboard (2million words, transcribed phone conversations)

Automatic annotation followed by hand corrections

[36]

How difficult is POS tagging in English?

- Roughly 15% of word types are ambiguous
    - Hence 85% of word types are unambiguous
    - *Janet* is always PROPN, *hesitantly* is always ADV
- But those 15% tend to be very common
- So ~60% of word tokens are ambiguous
- E.g., *back*

    earnings growth took a back/ADJ seat
    a small building in the back/NOUN
    a clear majority of senators back/VERB the bill
    enable the country to buy back/PART debt
    I was twenty-one back/ADV then

www.jads.nl

Jheronimus Academy of Data Science

POS tagging performance in English

- How many tags are correct? (Tag accuracy)
  - About 97%
    - Hasn't changed in the last 10+ years
    - HMMs, CRFs, BERT perform similarly .
    - Human accuracy about the same
- But baseline is 92%!
  - Baseline is performance of stupidest possible method
    - "Most frequent class baseline" is an important baseline for many tasks
      - Tag every word with its most frequent tag
      - (and tag unknown words as nouns)
  - Partly easy because
    - Many words are unambiguous

Sources of information for POS tagging

□□□□□ □□□□ □□□□ □□□ □□□□

AUX/NOUN/VERB? NOUN/VERB?

- Prior probabilities of word/tag
  - "will" is usually an AUX
- Identity of neighboring words
  - "the" means the next word is probably not a verb
- Morphology and word shape:
  - Prefixes unable: un- → ADJ
  - Suffixes importantly: -ly → ADJ
  - Capitalization Janet: CAP → PROPN

www.jads.nl

JADS Jheronimus Academy of Data Science

Standard algorithms for POS tagging

- Supervised Machine Learning Algorithms:
  - Hidden Markov Models (HMM)
  - Conditional Random Fields (CRF), Maximum Entropy Markov Models (MEMM)
  - Neural sequence models (RNNs or Transformers)
  - Large Language Models (like BERT), fine-tuned
- All required a hand-labeled training set, all about equal performance (97% on English)
- All make use of information sources we discussed
  - Via human created features: HMMs and CRFs
  - Via representation learning: Neural Language Models (LM)

Examples of NLP tasks: Classification

Based on slides from Jurafsky et al. (2020)

41

Jheronimus Academy of Data Science

Examples of text classification

- Sentiment analysis
- Spam detection
- Authorship identification
- Language Identification
- Assigning subject categories, topics, or genres
- …

Is this spam?

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

Greats News!

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

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

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

© Stanford University. All Rights Reserved.

[43]

www.jads.nl

Jheronimus Academy of Data Science

Positive or negative movie review?

- \+ *...zany characters and richly applied satire, and some great plot twists*
- − *It was pathetic. The worst part about it was the boxing scenes...*
- \+ *... awesome caramel sauce and sweet toasty almonds. I love this place!*
- − *...awful pizza and ridiculously overpriced...*

Positive or negative movie review?

+ *...zany characters and richly applied satire, and some great plot twists*

− *It was pathetic. The worst part about it was the boxing scenes...*

+ *... awesome caramel sauce and sweet toasty almonds. I love this place!*

− *...awful pizza and ridiculously overpriced...*

www.jads.nl

Jheronimus Academy of Data Science

47

Why sentiment analysis?

- *Movie*: is this review positive or negative?
- *Products*: what do people think about the new iPhone?
- *Public sentiment*: how is consumer confidence?
- *Politics*: what do people think about this candidate or issue?
- *Prediction*: predict election outcomes or market trends from sentiment

Text Classification: definition

- *Input*:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
- *Output*: a predicted class $c \in C$

[50]

www.jads.nl

Classification Methods: Hand-coded rules

- Rules based on combinations of words or other features
  - spam: black-list-address OR (“dollars” AND “you have been selected”)
- Accuracy can be high
  - If rules carefully refined by expert
- Building and maintaining these rules is expensive

www.jads.nl

Jheronimus Academy of Data Science

Classification Methods: Supervised Machine Learning

- Input:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

- Output:
  - a learned classifier $\gamma: d \rightarrow c$

Classification Methods: Supervised Machine Learning

Any kind of classifier

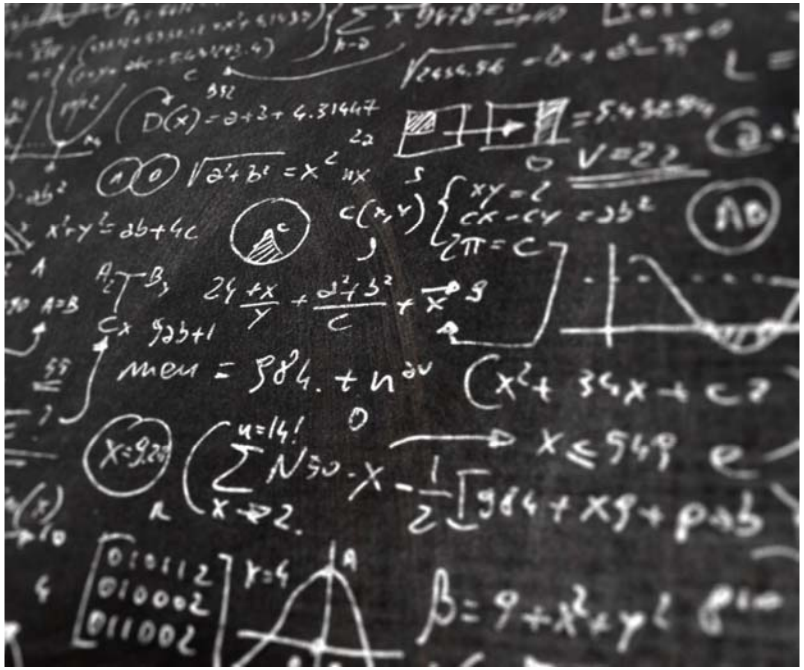
- Naïve Bayes
- Logistic regression
- Neural networks
- k-Nearest Neighbors
- …

Naive Bayes Classifier

Jheronimus
Academy
of Data Science

Naive Bayes Intuition

- Simple ("naive") classification method based on Bayes rule
- Relies on very simple representation of document
  - Bag of words



[55]

www.jads.nl

Jheronimus Academy of Data Science

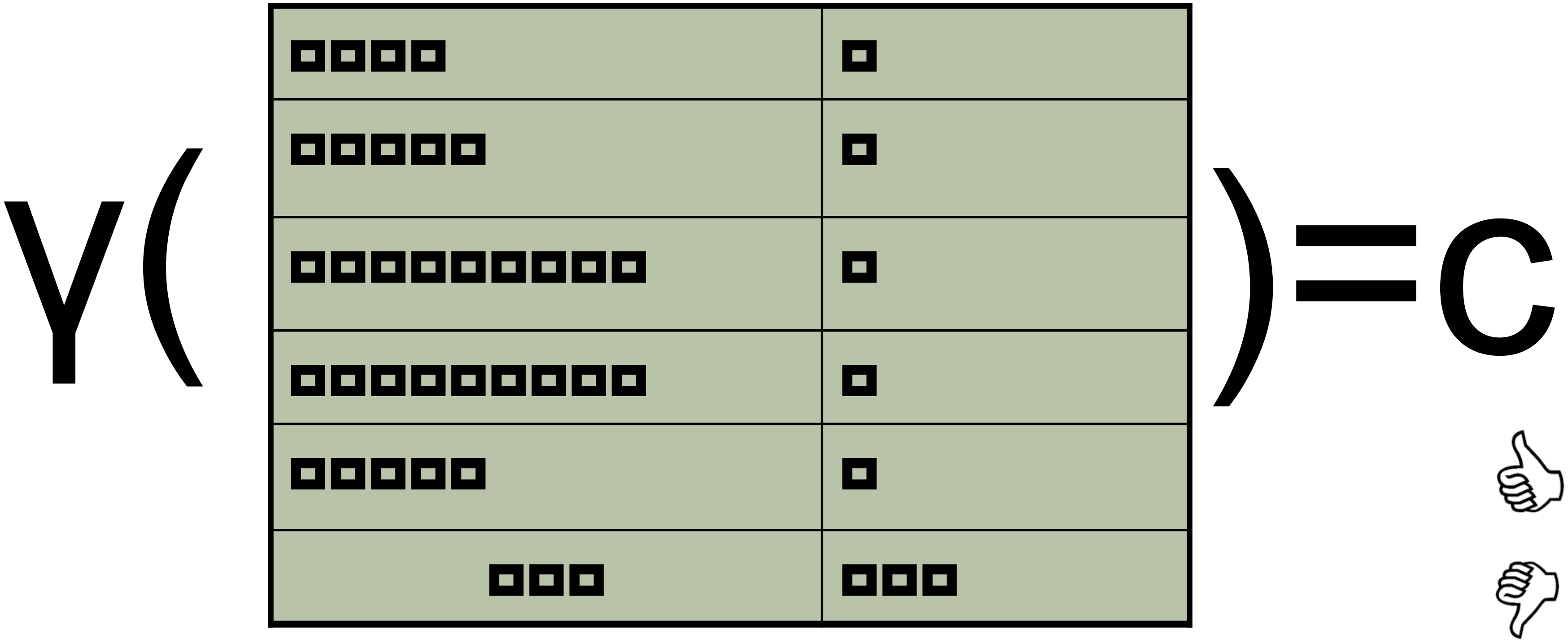
The Bag of Words Representation

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!

6
5
4
3
3
2
1
1
1
1
1
1
1
1
1
1
1
1
...

www.jads.nl

The bag of words representation



www.jads.nl

Jheronimus Academy of Data Science

Bayes' Rule Applied to Documents and Classes

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

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

Naïve Bayes Classifier (I)

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

MAP is "maximum a posteriori" = most likely class

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

Bayes Rule

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

Dropping the denominator

Naïve Bayes Classifier (II)

"Likelihood" "Prior"

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

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

Document d represented as features $x_1 \ldots x_n$

www.jads.nl

Naïve Bayes Classifier (III)

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

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

How often does this class occur?

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

We can just count the relative frequencies in a corpus

www.jads.nl

Multinomial Naïve Bayes Independence Assumptions

- **Bag of Words assumption**: Assume position doesn't matter

- **Conditional Independence**: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

www.jads.nl

Jheronimus Academy of Data Science

Multinomial Naïve Bayes Classifier

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

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

www.jads.nl

Jheronimus Academy of Data Science

Applying Multinomial Naïve Bayes Classifiers to Text Classification

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

www.jads.nl

Problems with multiplying lots of probs

There's a problem with this:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Multiplying lots of probabilities can result in floating-point underflow!

.0006 * .0007 * .0009 * .01 * .5 * .000008….

Idea: Use logs, because log(*ab*) = log(*a*) + log(*b*)

We sum up logs of probabilities instead of multiplying probabilities!

www.jads.nl

Jheronimus Academy of Data Science

Do all computations in log space

Instead of this: $$c_{NB} = \operatorname*{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

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

Notes:

1) Taking log doesn't change the ranking of classes!
   The class with highest probability also has highest log probability!
2) It's a linear model:
   Just a max of a sum of weights: a **linear** function of the inputs
   So naïve bayes is a **linear classifier**

Sec.13.3

Learning the Multinomial Naïve Bayes Model

- Estimate probabilities from frequencies in the data

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

www.jads.nl

Parameter estimation

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears among all words in documents of class c_j

- Create mega-document for class *j* by concatenating all docs in this class
  - Use frequency of *w* in mega-document

www.jads.nl

Problem with counting

- What if we have seen no training documents with the word ***fantastic*** and classified in the class **positive** (***thumbs-up)***?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic", positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

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

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

www.jads.nl

Jheronimus Academy of Data Science

Multinomial Naïve Bayes: Learning

From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ all docs with class = c_j

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

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

www.jads.nl

Unknown words

- What about unknown words
  - that appear in our test data
  - but not in our training data or vocabulary?
- We **ignore** them
  - Remove them from the test document!
  - Pretend they weren't there!
  - Don't include any probability for them at all!
- Why don't we build an unknown word model?
  - It doesn't help: knowing which class has more unknown words is not generally helpful!

Stop words

Some systems ignore stop words

- **Stop words:** very frequent words like *the* and *a*.
  - Sort the vocabulary by word frequency in training set
  - Call the top 10 or 50 words the **stopword list**.
  - Remove all stop words from both training and test sets
    - As if they were never there!

But removing stop words doesn't usually help

- In practice most NB algorithms use **all** words and **don't** use stopword lists

[73]

www.jads.nl

Jheronimus Academy of Data Science

Illustrative example

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

74

A worked example with add-1 smoothing

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable ~~with~~ no fun

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5
P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

$$\boxed{P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Optimizing for sentiment analysis

For tasks like sentiment, word **occurrence** seems to be more important than word **frequency**.

- The occurrence of the word *fantastic* tells us a lot
- The fact that it occurs 5 times may not tell us much more

Binary multinominal naïve bayes, or **binary NB**

- Clip our word counts at 1

[76]

www.jads.nl

Jheronimus Academy of Data Science

Binary Multinomial Naïve Bayes: learning

- From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ all docs with class = c_j

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

- Calculate $P(w_k \mid c_j)$ terms
  - Remove duplicates in each doc:
    - For each word type w in doc_j
      - Retain only a single instance of w
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

www.jads.nl

Binary Multinomial Naïve Bayes on a test document *d*

- First remove all duplicate words from *d*
- Then compute NB using the same equation:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(w_i \mid c_j)$$

Binary multinominal naïve Bayes – example

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

Binary multinominal naïve Bayes – example

Four original documents:

– it was pathetic the worst part was the boxing scenes
– no plot twists or great scenes
\+ and satire and great plot twists
\+ great scenes great film

	NB Counts +	NB Counts −
and	2	0
boxing	0	1
film	1	0
great	3	1
it	0	1
no	0	1
or	0	1
part	0	1
pathetic	0	1
plot	1	1
satire	1	0
scenes	1	2
the	0	2
twists	1	1
was	0	2
worst	0	1

www.jads.nl

Binary multinominal naïve Bayes – example

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- − it was pathetic the worst part boxing scenes
- − no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts +	NB Counts −
and	2	0
boxing	0	1
film	1	0
great	3	1
it	0	1
no	0	1
or	0	1
part	0	1
pathetic	0	1
plot	1	1
satire	1	0
scenes	1	2
the	0	2
twists	1	1
was	0	2
worst	0	1

Binary multinominal naïve Bayes – example

Four original documents:

− it was pathetic the worst part was the boxing scenes
− no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film

After per-document binarization:

− it was pathetic the worst part boxing scenes
− no plot twists or great scenes
+ and satire great plot twists
+ great scenes film

	NB Counts +	NB Counts −	Binary Counts +	Binary Counts −
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Counts can still be 2! Binarization is within-doc!

www.jads.nl

Jheronimus Academy of Data Science

Model evaluation

- As discussed before…
- Based on confusion matrix
- Use metrics such as accuracy, precision, recall, sensitivity, kappa, F-measure
- Confusion matrix works also for more than two classes
  - E.g. for 3-class classification

gold labels

system output	urgent	normal	spam	
urgent	8	10	1	$\textbf{precision}_u = \frac{8}{8+10+1}$
normal	5	60	50	$\textbf{precision}_n = \frac{60}{5+60+50}$
spam	3	30	200	$\textbf{precision}_s = \frac{200}{3+30+200}$
	$\textbf{recall}_u = \frac{8}{8+5+3}$	$\textbf{recall}_n = \frac{60}{10+60+30}$	$\textbf{recall}_s = \frac{200}{1+50+200}$	

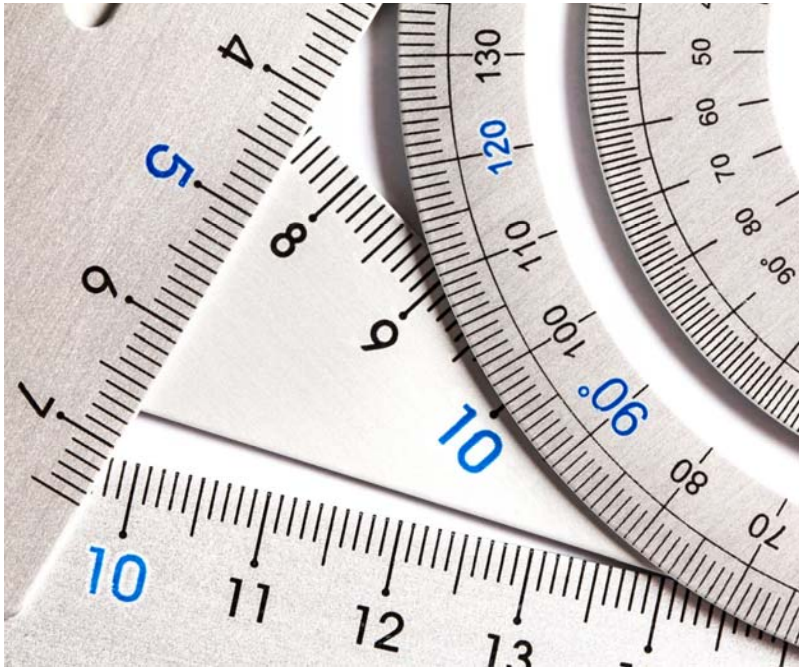
[83]

www.jads.nl

How to combine metrics from multiple classes into one

- Macro-averaging:
  - compute the performance for each class, and then average over classes

- Micro-averaging:
  - collect decisions for all classes into one confusion matrix
  - compute precision and recall from that table.



[84]

www.jads.nl

Jheronimus Academy of Data Science

Macro-averaging and Micro-averaging

system output \ *gold labels*	urgent	normal	spam	
urgent	8	10	1	$\textbf{precision}_u = \frac{8}{8+10+1}$
normal	5	60	50	$\textbf{precision}_n = \frac{60}{5+60+50}$
spam	3	30	200	$\textbf{precision}_s = \frac{200}{3+30+200}$
	$\textbf{recall}_u = \frac{8}{8+5+3}$	$\textbf{recall}_n = \frac{60}{10+60+30}$	$\textbf{recall}_s = \frac{200}{1+50+200}$	

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

www.jads.nl

Naïve Bayes is not so naïve

- Very Fast, low storage requirements
- Work well with very small amounts of training data
- 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 little data
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification
  - **Many other classifiers that give better accuracy**

Slide from Chris Manning

Summary

- Machine learning algorithms for text typically use a Vector Space Model
  - Bag of words
  - N-grams
  - Word embeddings
- POS tagging is useful for various NLP tasks
- Text classification is an important NLP application
  - Supervised classifiers, e.g. Naïve Bayes
  - Conventional ML text classifiers use BOW representation

[87]

www.jads.nl

Jheronimus Academy of Data Science