Computational Linguistics

Statistical NLP

Aurélie Herbelot

Centre for Mind/Brain Sciences
University of Trento

2016

Table of Contents

Introduction

Statistical NLP

- Most modern NLP applications are statistical in nature.
- Language is highly ambiguous. Using statistics allows us to guess what the best interpretation of a word/sentence/document might be.
- Learning from statistics is arguably closer to the way humans learn languages. (Important for AI applications.)
- A statistical system can easily be adapted in response to language change and is more robust to noise.

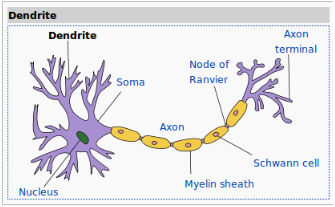
Language is ambiguous

- Recap from last week: language is ambiguous at the lexical and structural level.
  - *smoke*
  - *Kim saw the woman with the telescope.*
- Gathering statistics can help us decide which interpretation is most plausible.

The AI/cognitive aspect of statistics

- Hebb's rule:

  When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.



The AI/cognitive aspect of statistics

- Children primarily learn their language through natural exposure to it.
- Unclear whether some aspects of ur-syntax are encoded in the new-born's brain.
- The lexicon and syntax of a particular language, as well as the language's relation to the world is learnt by hearing.
- A good language understanding system should learn as humans do.

Language ain't a grammar

- Manually built grammars fix language in a certain state.
- They ignore language change, speaker errors, etc.
  - *I don't have no time for this!*
  - *can u cum tmrw cos tue i'm busy*
  - *Prefect! See you!*
  - *Il ne croyait pas qu'il était rentré.*

Where to get stats from?

- **They're already there:** for instance, we want to know what kind of words appear next to a particular term (distributional semantics).
- **Annotations:** we ask humans to label some text according to predefined rules:
  - Experts (e.g. for parsing).
  - Non-experts (to get more intuitive annotations).

Building a statistical NLP application

- Choose your data carefully (according to the task of interest).
- Produce or gather annotation (according to the task of interest).
- Randomly split the annotated data into training, validation and test set.
    - The training data is to ‘learn the rules’.
    - The validation data is to tune parameters (if needed).
    - The test data is the unknown set which gives system performance ‘in the real world’.
- Choose appropriate features.
- Learn!

The statistical NLP pipeline

- The typical NLP pipeline includes the following components:
  - A sentence splitter.
  - A tokeniser.
  - A lemmatiser.
  - A part-of-speech tagger.
  - A syntactic/semantic parser.
  - Others: co-reference resolution, Named Entity recognition, etc.

Sentence splitting: the need for statistics

- Sentence splitting: the task of splitting a text into component sentences.
- A rule-based approach: split on ., ? or !
- This won't work:
  - *I don't know what to do...*
  - *The painting has sold for 1.2M Euros.*
  - *The exclamation mark (!) is used to indicate surprise.*
  - *(But don't tell him, it's a surprise!)* Anyway...
  - *She asked 'What?' in an indignant voice.*
  - *i can't believe u did that ur crazy*
- Statistics can tell us that, when followed by two other dots, a dot is part of an ellipsis. Or that when preceded and followed by a digit, it is part of a number.

Sentence splitting: the need for statistics

- Sentence splitting: the task of splitting a text into component sentences.
- A rule-based approach: split on ., ? or !
- This won't work:
  - *I don't know what to do.|..*
  - *The painting has sold for 1.2M Euros.*
  - *The exclamation mark (!) is used to indicate surprise.*
  - *(But don't tell him, it's a surprise!)* Anyway...
  - *She asked 'What?' in an indignant voice.*
  - *i can't believe u did that ur crazy*
- Statistics can tell us that, when followed by two other dots, a dot is part of an ellipsis. Or that when preceded and followed by a digit, it is part of a number.

Sentence splitting: the need for statistics

- Sentence splitting: the task of splitting a text into component sentences.
- A rule-based approach: split on ., ? or !
- This won't work:
  - *I don't know what to do...*
  - *The painting has sold for 1.|2M Euros.*
  - *The exclamation mark (!) is used to indicate surprise.*
  - *(But don't tell him, it's a surprise!)* Anyway...
  - *She asked 'What?' in an indignant voice.*
  - *i can't believe u did that ur crazy*
- Statistics can tell us that, when followed by two other dots, a dot is part of an ellipsis. Or that when preceded and followed by a digit, it is part of a number.

Sentence splitting: the need for statistics

- Sentence splitting: the task of splitting a text into component sentences.
- A rule-based approach: split on ., ? or !
- This won't work:
    - *I don't know what to do...*
    - *The painting has sold for 1.2M Euros.*
    - *The exclamation mark (!|) is used to indicate surprise.*
    - *(But don't tell him, it's a surprise!)* Anyway...
    - *She asked 'What?' in an indignant voice.*
    - *i can't believe u did that ur crazy*
- Statistics can tell us that, when followed by two other dots, a dot is part of an ellipsis. Or that when preceded and followed by a digit, it is part of a number.

Sentence splitting: the need for statistics

- Sentence splitting: the task of splitting a text into component sentences.
- A rule-based approach: split on ., ? or !
- This won't work:
  - *I don't know what to do...*
  - *The painting has sold for 1.2M Euros.*
  - *The exclamation mark (!) is used to indicate surprise.*
  - *(But don't tell him, it's a surprise!|)* Anyway...
  - *She asked 'What?' in an indignant voice.*
  - *i can't believe u did that ur crazy*
- Statistics can tell us that, when followed by two other dots, a dot is part of an ellipsis. Or that when preceded and followed by a digit, it is part of a number.

Sentence splitting: the need for statistics

- Sentence splitting: the task of splitting a text into component sentences.
- A rule-based approach: split on ., ? or !
- This won't work:
  - *I don't know what to do...*
  - *The painting has sold for 1.2M Euros.*
  - *The exclamation mark (!) is used to indicate surprise.*
  - *(But don't tell him, it's a surprise!)* Anyway...
  - *She asked 'What?|' in an indignant voice.*
  - *i can't believe u did that ur crazy*
- Statistics can tell us that, when followed by two other dots, a dot is part of an ellipsis. Or that when preceded and followed by a digit, it is part of a number.

Tokenising

- Given a sequence of characters, break it into *token* components (roughly-speaking, into words).
- Similar issues as with sentence splitting. What belongs to a word and what doesn't?
  - *doesn't*: apostrophe in the word?
  - *'token'*: quote marks *not* in the word.
  - **M*A*S*H**: asterisks part of the word.
  - *good-looking*: split or don't split?
  - *New York*: don't split!
  - ...

Lemmatisation

- Return the base form (lemma) of a particular token: *walks* –> *walk*, *cats* –> cat, etc.
- This reduces the vocabulary and is helpful in cases where the data is sparse (all words with roughly the same meaning are grouped together).
- Problems:
    - *camping* –> *camp*?
    - *better* –> *good*?
      cf. *She got the better of me.*

Part-of-speech tagging

- For each token, indicate its part-of-speech (POS):
  - NNS: plural noun
  - DT: determiner
  - VBZ: verb, 3rd person singular
  - ...
- The probability of a particular POS is dependent on the probability of the POS of the previous tokens: if I am fairly sure that token X_n is a determiner, there is a higher probability that X_{n+1} is a noun or an adjective. We learn this statistically.

A syntactic/semantic parser (last week!)

S	$\rightarrow$	$NP\ VP$	1.0
VP	$\rightarrow$	$VP\ PP$	0.7
VP	$\rightarrow$	$V\ NP$	0.5
VP	$\rightarrow$	$eats$	0.1
PP	$\rightarrow$	$P\ NP$	0.8
NP	$\rightarrow$	$Det\ N$	0.7
NP	$\rightarrow$	$NP\ PP$	0.2
NP	$\rightarrow$	she	0.1
NP	$\rightarrow$	$cake$	0.1
V	$\rightarrow$	$eats$	0.1
P	$\rightarrow$	$with$	0.2
N	$\rightarrow$	$fork$	0.1
Det	$\rightarrow$	a	0.2

	1	2	3	4	5	6
1	**NP**	**V**,VP	**NP**	**P**	**Det**	**N**
2	S	**VP**			**NP**	
3	S			**PP**		
4						
5		**VP**				
6	**S**					
	she	eats	cake	with	a	fork

(she(eats (cake))(with(a (fork))))

$P(T) = 0.1 * 0.1 * 0.1 * 0.2 * 0.2 * 0.1 * 0.5 * 0.7 * 0.8 * 0.7 * 1.0 =$

$7.84.10^{-}7$

A syntactic/semantic parser (last week!)

S	$\rightarrow$	$NP\ VP$	1.0
VP	$\rightarrow$	$VP\ PP$	0.7
VP	$\rightarrow$	$V\ NP$	0.5
VP	$\rightarrow$	*eats*	0.1
PP	$\rightarrow$	$P\ NP$	0.8
NP	$\rightarrow$	$Det\ N$	0.7
NP	$\rightarrow$	$NP\ PP$	0.2
NP	$\rightarrow$	*she*	0.1
NP	$\rightarrow$	*cake*	0.1
V	$\rightarrow$	*eats*	0.1
P	$\rightarrow$	*with*	0.2
N	$\rightarrow$	*fork*	0.1
Det	$\rightarrow$	*a*	0.2

	1	2	3	4	5	6
1	**NP**	**V**,VP	**NP**	**P**	**Det**	**N**
2	S				**NP**	
3	S			**PP**		
4			**NP**			
5		**VP**				
6	**S**					
	she	eats	cake	with	a	fork

(she (eats)(cake(with(a(fork)))))

$P(T) = 0.1 * 0.1 * 0.1 * 0.2 * 0.2 * 0.1 * 0.7 * 0.8 * 0.2 * 0.5 * 1.0 =$

$2.24.10^{-}7$

Statistical applications: document classification


sign in | become a supporter | subscribe | search | jobs | dating | more | International

theguardian

UK world sport football opinion culture business lifestyle fashion environment tech travel | browse all sections

home › world › americas | asia australia africa middle east cities development europe US

Haiti

Hurricane Matthew leaves Haiti facing humanitarian crisis - UN

Eleven confirmed dead, thousands forced into shelters and water in short supply as storm sweeps across Caribbean

Sam Jones, Nicky Woolf and agencies

Wednesday 5 October 2016 12.10 BST

Most popular

Next Generation 2016: 60 of the best young talents in world football


sign in | become a supporter | subscribe | search | jobs | dating | more | International

theguardian

UK world sport football opinion culture business lifestyle fashion environment tech travel | browse all sections

home › tech

Yahoo

Yahoo 'secretly monitored emails on behalf of the US government'

Company complied with a classified directive, scanning hundreds of millions of Yahoo Mail accounts at the behest of NSA or FBI, say former employees

Nicky Woolf in San Francisco and agencies

@nickywoolf

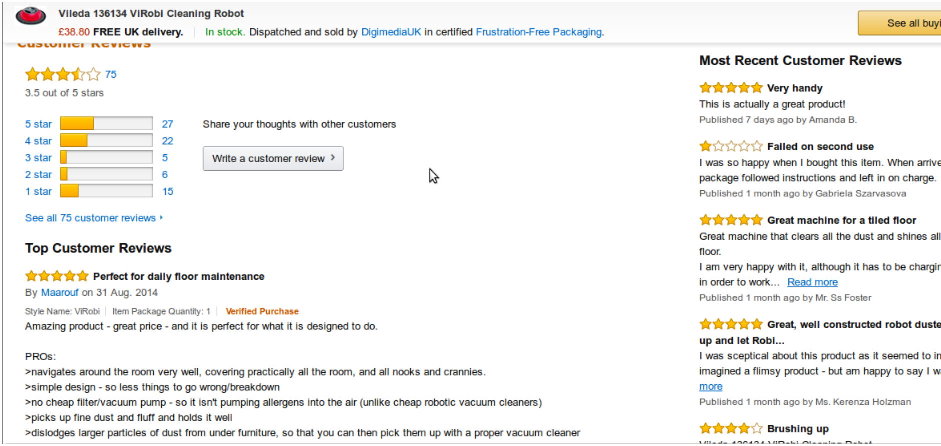
Wednesday 5 October 2016 07.58 BST

Most popular

Next Generation 2016: 60 of the best young talents in world football

Waiting for ophan.theguardian.com...

Statistical applications: sentiment analysis



Statistical applications: authorship attribution


www.thetimes.co.uk/tto/arts/books/article3816183.ece

Search

JK Rowling unmasked as author of bestselling crime novel

Article | Analysis: Alexi Mostrous

Behind the story:

The Cuckoo's Calling by Robert Galbraith

London never looked emptier or more jaded than in this fine debut novel starring a damaged hero with a differ...

Last updated at May 11 2013

Post a comment

Mixed reviews but good sales for Rowling's The Casual Vacancy

The Casual Vacancy is greeted with a mixture of critical praise and derision by reviewers across the globe

Last updated at September 27 2012

Post a comment

Harry Potter and the magica

By continuing to use the site, you agree to the use of cookies. You can change this and find out more by following this link.

Accept Cookies

Statistical applications: modelling the brain

http://news.stanford.edu/news/2013/march/images/neuroimage_news.jpg

Naive Bayes

Probabilities: crash reminder

- $P(A)$: the frequency of event A, relative to all other possible events, given an experiment repeated an *infinite* number of times.
- $P(A|B)$: the probability of A given B.
- Assumption of independence: $P(A \cap B) = P(A) * P(B)$ if A and B are independent.

Probabilistic classification

- We want to model the conditional probability of output labels **y** given input **x**.
- For instance: model the probability of a film review being positive ($y = 1$ or $y = 0$) given the words in the review ($x = \{$ ... the worst action film ... $\}$).
- We want to evaluate $P(y|x)$ and find $\mathrm{argmax}_y P(y|x)$.

Bayes' Rule

- We can model $P(y|x)$ through Bayes' rule:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \tag{1}$$

- Finding the argmax means using the following equivalence:

$$\underset{y}{\operatorname{argmax}}\, P(y|x) \propto \underset{y}{\operatorname{argmax}}\, P(x|y)P(y) \tag{2}$$

Naive Bayes Model

- Let $\Theta(x)$ be a set of features such that $\Theta(x) = \theta_1(x), \theta_2(x), ..., \theta_n(x)$.
- $P(x|y) = P(\theta_1(x), \theta_2(x), ..., \theta_n(x)|y)$.
- We use the naive bayes assumption of conditional independence: $P(\theta_1(x), \theta_2(x), ..., \theta_n(x)|y) = \prod_i P(\theta_i(x)|y)$
- $P(x|y)P(y) = P(y)\prod_i P(\theta_i(x)|y)$

Maximum Likelihood Estimates

- $P(y)$ is the relative frequency of y in the training data.
- $P(\theta_i(x)|y)$ is the ratio between the frequency of $(\theta_i(x), y)$ and the frequency of y in the training data.

Naive Bayes Example

- Let's say your mailbox is organised as follows:
    - Work
        - Eva
        - Angeliki
        - Abhijeet
    - Friends
        - Tim
        - Jim
        - Kim
- You want to automatically file new emails according to their topic (work or friends).

Document classification

- Classify document into one of two classes: *work* or *friends*.
  $y = [0, 1]$, where 0 is for *work* and 1 is for *friends*.
- Use words as features (under the assumption that the meaning of the words will be indicative of the meaning of the documents, and thus its topic).
  $\theta_i(x) = w_i$
- We have one feature per word in our vocabulary V (the 'vocabulary' being the set of unique words in all texts encountered in training).

Some training emails

- E1: “Shall we go climbing at the weekend?”
  friends
- E2: “The composition function can be seen as one-shot learning.”
  work
- E3: “We have to finish the code at the weekend.”
  work

- $V = \{$ shall we go climbing at the weekend ? composition function can be seen as one-shot learning . have to finish code $\}$

Some training emails

- E1: “Shall we go climbing at the weekend?”
  friends
- E2: “The composition function can be seen as one-shot learning.”
  work
- E3: “We have to finish the code at the weekend.”
  work

- $\Theta(x) = \{$ shall we go climbing at the weekend ? composition function can be seen as one-shot learning . have to finish code $\}$

Some training emails

- E1: “Shall we go climbing at the weekend?”
  friends
- E2: “The composition function can be seen as one-shot learning.”
  work
- E3: “We have to finish the code at the weekend.”
  work

- $P(\Theta(x)|y = 0) = \{$ (shall,0) (we,0.5) (go,0) (climbing,0) (at,0.5) (the,0.75) (weekend,0.5) (?,0) (composition,1) (function,1) (can,1) (be,1) (seen,1) (as,1) (one-shot,1) (learning,1) (.,1) (have,1) (to,1) (finish,1) (code,1) $\}$

Some training emails

- E1: “Shall we go climbing at the weekend?”
  friends
- E2: “The composition function can be seen as one-shot learning.”
  work
- E3: “We have to finish the code at the weekend.”
  work

- $P(\Theta(x)|y = 1) = \{$ (shall,1) (we,0.5) (go,1) (climbing,1) (at,0.5) (the,0.25) (weekend,0.5) (?,1) (composition,0) (function,0) (can,0) (be,0) (seen,0) (as,0) (one-shot,0) (learning,0) (.,0) (have,0) (to,0) (finish,0) (code,0) $\}$

Prior class probabilities

- $P(0) = \frac{f(doctopic=0)}{f(alldocs)} = \frac{2}{3} = 0.66$
- $P(1) = \frac{f(doctopic=1)}{f(alldocs)} = \frac{1}{3} = 0.33$

A new email

- E4: “When shall we finish the composition code?”

- We ignore unknown words: (*when*).
- $V = \{$ shall we finish the composition code ? $\}$
- We want to solve:

$$\operatorname*{argmax}_y P(y|\Theta(x)) \propto \operatorname*{argmax}_y P(\Theta(x)|y)P(y) \tag{3}$$

Testing $y = 0$

$$
\begin{aligned}
& P(\Theta(x)|y) \\
= \quad & P(shall|y=0) * P(we|y=0) * P(finish|y=0) * \\
& P(the|y=0) * P(composition|y=0) * \\
& P(code|y=0) * P(?|y=0) \\
= \quad & 0 * 0.5 * 1 * 0.75 * 1 * 1 * 0 \\
= \quad & 0
\end{aligned}
$$

Oops.......

Smoothing

- When something has probability 0, we don't know whether that is because the probability is *really* 0, or whether the training data was simply 'incomplete'.
- Smoothing: we add some tiny probability to unseen events, just in case...
- Additive/Laplacian smoothing:

$$P(e) = \frac{f(e)}{\sum_{e'} f(e')} \rightarrow P(e) = \frac{f(e) + \alpha}{\sum_{e'} f(e') + \alpha} \tag{4}$$

Recalculating training probabilities...

- E1: "Shall we go climbing at the weekend?"
  friends
- E2: "The composition function can be seen as one-shot learning."
  work
- E3: "We have to finish the code at the weekend."
  work

- **Examples:**
  - $P(\mathit{the}|y=0) = \frac{3+0.01}{4+0.01} \approx 0.75$
  - $P(\mathit{climbing}|y=0) = \frac{0+0.01}{1+0.01} \approx 0.01$

Testing $y = 0$ (*work*)

$$
\begin{aligned}
& P(\Theta(x)|y) \\
= \quad & P(shall|y=0) * P(we|y=0) * P(finish|y=0) * \\
& P(the|y=0) * P(composition|y=0) * \\
& P(code|y=0) * P(?|y=0) \\
= \quad & 0.01 * 0.5 * 1 * 0.75 * 1 * 1 * 0.01 \\
= \quad & 3.75 * 10^{-5}
\end{aligned}
$$

$$
\begin{aligned}
& P(\Theta(x)|y)P(y) \\
= \quad & 3.75 * 10^{-5} * 0.66 \\
= \quad & 2.475 * 10^{-5}
\end{aligned}
$$

Testing $y = 1$ (*friends*)

$$
\begin{aligned}
& P(\Theta(x)|y) \\
= \quad & P(shall|y=1) * P(we|y=1) * P(finish|y=1) * \\
& P(the|y=1) * P(composition|y=1) * \\
& P(code|y=1) * P(?|y=1) \\
= \quad & 1 * 0.5 * 0.01 * 0.25 * 0.01 * 0.01 * 1 \\
= \quad & 1.25 * 10^{-7}
\end{aligned}
$$

$$
\begin{aligned}
& P(\Theta(x)|y)P(y) \\
= \quad & 1.25 * 10^{-7} * 0.33 \\
= \quad & 4.125 * 10^{-8}
\end{aligned}
$$

The issue of feature selection

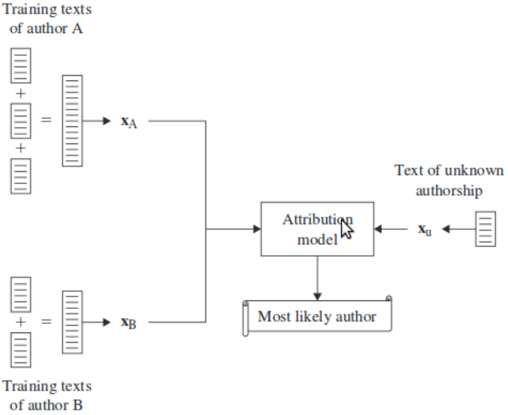
Authorship attribution

- Your mailbox is organised as follows:
  - Work
    - Eva
    - Angeliki
    - Abhijeet
  - Friends
    - Tim
    - Jim
    - Kim
- How different are the emails from Eva and Abhijeet? From Tim and Jim?

Authorship attribution

- The task of deciding *who* has written a particular text.
- Useful for historical, literature research. (Are those letters from Van Gogh?)
- Used in forensic linguistics; by companies who want to sell you things you don't want; by companies who sell you to companies who sell you things you don't want...
- Interesting from the point of view of feature selection.

Basic architecture of authorship attribution



From Stamatatos (2009). *A Survey of Modern Authorship Attribution Methods.*

Choosing features

- Which features might be useful in authorship attribution?
  - Stylistic: does the person tend to use lots of adverbs? To hedge their statements with modals?
  - Lexical: *what* does the person talk about?
  - Syntactic: does the person prefer certain syntactic patterns to others?
  - Other: does the person write smileys with a nose or without? :-) :)

Stylistic features

- The oldest types of features for authorship attribution (Mendenhall, 1887).
- Word length, sentence length... (Are you pompous? Complicated?)
- Vocabulary richness (type/token ratio). But: dependent on text length. The size of vocabulary increases rapidly at the beginning of a text and then decreases.

Lexical features

- The most widely used feature in authorship attribution.
- A text is represented as a vector of word frequencies.
- This is then only a rough topical representation which disregard word order.
- N-grams combine the best of all words, encoding order and some lexical information (coming soon...)

Syntactic features

- Syntax is used largely unconsciously and is thus a good indicator of authorship.
- An author might keep using the same patterns (e.g. prefer passive forms to active ones).
- But producing good features relies on having a good parser...
- Partial solution: use shallow syntactic features, e.g. sequences of POS tags (DT JJ NN).

The case of emoticons

- Which ones are used? :-) :D :P ^_^
  - Indication of geographical provenance.
- How are they written? :-) or :)
  - Indication of age.
- Miscellaneous: how do you put a smiley at the end of a parenthesis?
  a) (cool! :)) b) (cool! :) c) (cool! :)) ...

Simple is best

- The best features for authorship attribution are often the simplest.
- Use of function words (prepositions, articles, punctuation) is usually more revealing than content words. They are mostly used unconsciously by authors.
- N-grams are a powerful and simple technique, relying on strings of characters:
  - unigrams: n, -, g, r, a, m
  - bigrams: n-, -g, gr, ra, am, ms
  - trigrams: n-g, -gr, gra, ram, ams

N-grams

- N-grams which is both robust to noise and captures various types of information, including:
  - frequency of various prepositions (*_in_*, *for_*);
  - use of punctuation (*;_an*);
  - abbreviations (*e_&_*);
  - even lexical features (*type*, *text*, *ment*).

An n-gram authorship attribution algorithm

- Collect all N-grams in training data: one frequency table per author.

```
def record_ngrams(filename):
  ngrams={}
  f=open(filename,'r')
  for l in f:
    l=l.rstrip('\n')
    for i in range(len(l)-n):
      ngram=l[i:i+n]
      if ngram in ngrams:
        ngrams[ngram]+=1
      else:
        ngrams[ngram]=1
  f.close()
  return ngrams
```

An n-gram authorship attribution algorithm

- Record frequencies over the whole training data in one single vocabulary.

```
def fill_vocab(author_ngrams):
    for ngram,freq in author_ngrams.items():
      if ngram in vocab:
        vocab[ngram]+=freq
      else:
        vocab[ngram]=freq
```

An n-gram authorship attribution algorithm

- Calculate probabilities (including smoothing) for each <ngram,author> pair.

```
#Calculate ngram probabilities for each author
for d in ngrams_dicts:
  for ngram,freq in d.items():
    d[ngram]=float(freq+alpha)/float(vocab[ngram]+alpha)
```

An n-gram authorship attribution algorithm

- Calculate probability of each author given ngrams of new text.

```
def naive_bayes(test_ngrams,author):
  nb=0
  for ngram,freq in test_ngrams.items():
    if ngram in ngrams_dicts[author]:
      for i in range(freq):
        nb=nb+math.log(ngrams_dicts[author][ngram])
    else:
      if ngram in vocab:
        for i in range(freq):
          nb=nb+math.log(float(alpha)/float(vocab[ngram]+alpha))
  print nb
```

Changing features

- The function extracting the features can be modified without affecting the rest of the code.

```
def record_ngrams(filename):
  ngrams={}
  f=open(filename,'r')
  for l in f:
    l=l.rstrip('\n')
    for i in range(len(l)-n):
      ngram=l[i:i+n]
      if ngram in ngrams:
        ngrams[ngram]+=1
      else:
        ngrams[ngram]=1
  f.close()
  return ngrams
```

A test

- A small test of the algorithm:
  - Download some books from Project Gutenberg.
  - Produce features for all books.
  - Do leave-one-out evaluation: use all books but one for training, one book for testing.
  - Do we return the correct author?

A test

- Four authors: Austen, Carroll, Grahame, Kipling.
- Retain one book: *Emma* by Jane Austen.
- Results:
  - n-grams: +1
  - words: +1
  - top 50 function words: +1

Function words in action

the	568	to	4051	the	3182	the	3363
and	294	the	4047	and	2125	and	2868
a	265	of	3554	of	1176	of	1361
to	241	and	3240	to	1142	to	1328
she	197	a	1889	a	1038	a	1238
of	189	her	1858	he	846	he	846
was	159	was	1795	in	642	in	799
in	153	in	1759	his	635	his	701
it	112	I	1740	that	559	was	584
her	92	that	1406	was	501	that	494
I	89	not	1337	I	470	on	480
as	80	she	1306	for	406	I	457
that	77	be	1191	is	402	with	450
you	72	his	1167	as	388	you	448
at	67	had	1126	with	365	as	388
with	65	as	1110	on	314	at	383
had	64	he	1036	had	283	for	380
on	55	with	994	not	265	it	380
all	50	for	982	they	262	had	336
down	44	it	942	at	261	all	297
out	44	you	937	all	260	they	292
for	40	have	813	him	247	be	288
The	40	is	781	but	234	him	274
into	39	at	736	up	231	The	266
up	37	on	637	have	221	up	246
very	37	by	612	you	195	by	231
be	36	but	605	it	192	very	210
She	35	my	583	The	186	so	190
or	34	were	542	out	177	out	184
they	32	so	518	when	168	or	175
like	32	all	511	be	166	into	172

Fight back: authorship obfuscation

- Obfuscation is hard, and that is a statistical problem.
- A lexical strategy: replace words by synonyms. But:
  - *parish* –> *community*?? (Semantic issue.)
  - *all* –> *every*?? (Grammaticality and typicality issue.)
- The problem is that the hearer/reader of the text has certain general expectations about what a text should look like. Often, lexical obfuscation results in something that a) doesn't look human; b) has changed the original meaning of the text.

Evaluation

Precision and recall: recap

	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

- Precision: $\frac{TP}{TP+FP}$
- Recall: $\frac{TP}{TP+FN}$
- Accuracy: $\frac{TP+TN}{TP+FN+FP+TN}$
- F-score: $\frac{2 * precision * recall}{recall + precision}$

Multiclass evaluation

- How to calculate precision/recall in the case of a multiclass problem (for instance, authorship attribution across 4 different authors).
- Calculate precision e.g. for class A by collapsing all other classes together.
-

	Predicted A	Predicted B	Predicted C
Actual A	TA	FB	FC
Actual B	FA	TB	FC
Actual C	FA	FB	TC

Multiclass evaluation

- How to calculate precision/recall in the case of a multiclass problem (for instance, authorship attribution across 4 different authors).
- Calculate precision e.g. for class A by collapsing all other classes together.
-

	Predicted A	Predicted $\overline{A}$
Actual A	TA = TA	F$\overline{A}$ = FB+FC
Actual $\overline{A}$	F$\overline{A}$ = FA_B+FA_C	T$\overline{A}$=TB+TC

Why is it not working?

- Bad data: the data we are learning from is not the right one for the task.
- Bad humans: the quality of the annotation is insufficient.
- Bad features: we didn't choose the right features for the task.
- Bad algorithm: the learning algorithm is too dumb.

Bad data

- It is not always very clear which data should be used for producing general language understanding systems.
- See Siri disasters:
  - Human: Siri, call me an ambulance.
  - Siri: From now on, I'll call you 'an ambulance'. Ok?

    http://www.siri-isms.com/siri-says-ambulance-533/
- Sometimes, the data is simply too small (data sparsity problem).

Bad humans

- The annotation process should be followed by a validation of the quality of the annotation.
- Measures of inter-annotator agreement. E.g. Cohen's Kappa: $\kappa = \frac{p_0 - p_e}{1 - p_e}$
- The assumption is that the more agreement we have, the better the data is.
- Magnitude guidelines: 0-0.20 is slight, 0.21-0.40 is fair, 0.41-0.60 is moderate, 0.61-0.80 is substantial, and 0.81-1 is almost perfect agreement.
- Sometimes useful to measure *intra*-annotator agreement!

Where does low agreement come from?

- The guidelines were bad. Compare:
  - How similar are *cat* and *dog*? (1-7)
  - Is *cat* more similar to *dog* or to *horse?*
- The task is hard: it requires access to knowledge that is normally unconscious, or too much interpretation.
  - Quantify the following with *no, few, some, most, all*:
    - ___ bathtubs are white
    - ___ trumpets are loud

Never trust humans to do what you want...

Predication type	Example	Prevalence
Principled	Dogs have tails	92%
Quasi-definitional	Triangles have three sides	92%
Majority	Cars have radios	70%
Minority characteristic	Lions have manes	64%
High-prevalence	Canadians are right-handed	60%
Striking	Pit bulls maul children	33%
Low-prevalence	Rooms are round	17%
False-as-existentials	Sharks have wings	5%

Table: Classes of generic statements with associated prevalence, as per Khemlani (2009).

Bad features

- The theory is wrong: the phenomenon we want to study is not dependent on the features we thought were important.
- A big aspect of recent neural network techniques is to build algorithms who can decide on their own which features are needed for a task.

Bad algorithm

- The algorithm is not powerful enough for the data.
- For instance, the independence assumption does not hold.
- Or the data is distributed in a complex way through the space...

Conclusion

Statistical NLP: you must get this right...

- The data must be representative of your task.
- You must have enough data.
- Your annotation guidelines must be precise, user-friendly, adapted to the task.
- Your annotation task shouldn't require too much interpretation.
- Your annotators must be serious!
- Your inter-annotator agreement measure must be right for the annotation.
- Your features must be right for the task.
- Your pre-processing tools must be accurate.
- Your algorithm should be powerful enough to separate your classes.
- You should write bug-free code :)
- ...