

# SIT330-770: Natural Language Processing

## Week 5 - Naïve Bayes and Sentiment Classification

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of  
Sci Eng & Built Env

[reda.bouadjenek@deakin.edu.au](mailto:reda.bouadjenek@deakin.edu.au)



# SIT330-770: Natural Language Processing

Week 5.1 - The Task of Text Classification

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



**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>

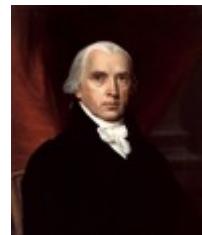
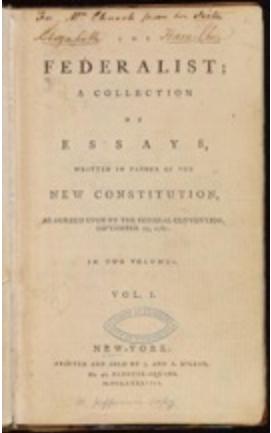
Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

© Stanford University. All Rights Reserved.

# Who wrote which Federalist papers?



- 1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.
- Authorship of 12 of the letters in dispute
- 1963: solved by Mosteller and Wallace using Bayesian methods

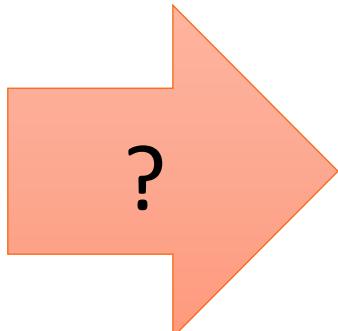


James Madison



Alexander Hamilton

# What is the subject of this medical article?



- **MeSH Subject Category Hierarchy**
  - Antagonists and Inhibitors
  - Blood Supply
  - Chemistry
  - Drug Therapy
  - Embryology
  - Epidemiology
  - ...

# 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...

# 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

- **Emotion:** brief organically synchronized ... evaluation of a major event
  - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood:** diffuse non-caused low-intensity long-duration change in subjective feeling
  - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances:** affective stance toward another person in a specific interaction
  - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes:** enduring, affectively colored beliefs, dispositions towards objects or persons
  - *liking, loving, hating, valuing, desiring*
- **Personality traits:** stable personality dispositions and typical behavior tendencies
  - *nervous, anxious, reckless, morose, hostile, jealous*

# Scherer Typology of Affective States



- **Emotion:** brief organically synchronized ... evaluation of a major event
  - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood:** diffuse non-caused low-intensity long-duration change in subjective feeling
  - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances:** affective stance toward another person in a specific interaction
  - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes:** enduring, affectively colored beliefs, dispositions towards objects or persons
  - *liking, loving, hating, valuing, desiring*
- **Personality traits:** stable personality dispositions and typical behavior tendencies
  - *nervous, anxious, reckless, morose, hostile, jealous*

- Sentiment analysis is the detection of **attitudes**
- Simple task we focus on in this chapter
  - Is the attitude of this text positive or negative?
- We return to affect classification in later chapters

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

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

- 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
- But building and maintaining these rules is expensive

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

# SIT330-770: Natural Language Processing

Week 5.2 - The Naive Bayes Classifier

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



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

# 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!



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

# The bag of words representation



Y( ) = C

|           |     |
|-----------|-----|
| seen      | 2   |
| sweet     | 1   |
| whimsical | 1   |
| recommend | 1   |
| happy     | 1   |
| ...       | ... |

Thumbs up icon

Thumbs down icon

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

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

# Naive Bayes Classifier (I)



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

MAP is “maximum a posteriori” = most likely class

$$= \operatorname{argmax}_{c \in C} \frac{P(d | c)P(c)}{P(d)}$$

Bayes Rule

$$= \operatorname{argmax}_{c \in C} P(d | c)P(c)$$

Dropping the denominator

## Naive Bayes Classifier (II)



"Likelihood"      "Prior"

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

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

Document d  
represented as  
features x<sub>1..n</sub>

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

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

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

How often does this class occur?

We can just count the relative frequencies in a corpus

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

- **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  $c$ .

$$P(x_1, \dots, x_n | c) = P(x_1 | c) \bullet P(x_2 | c) \bullet P(x_3 | c) \bullet \dots \bullet P(x_n | c)$$

# Multinomial Naive Bayes Classifier



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

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

# Applying Multinomial Naive Bayes Classifiers to Text Classification



positions  $\leftarrow$  all word positions in test document

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

# Problems with multiplying lots of probs



- There's a problem with this:

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

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'll sum logs of probabilities instead of multiplying probabilities!

# We actually do everything in log space



Instead of this:

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

This:

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

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 naive bayes is a **linear classifier**

# SIT330-770: Natural Language Processing

Week 5.3 - Naive Bayes: Learning

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

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

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

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

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

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

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

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

- 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



$$\begin{aligned}\hat{P}(w_i | c) &= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} \\ &= \frac{\text{count}(w_i, c) + 1}{\left( \sum_{w \in V} \text{count}(w, c) \right) + |V|}\end{aligned}$$

- 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  $P(w_k | c_j)$  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  $Text_j$

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

- 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!

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

# SIT330-770: Natural Language Processing

Week 5.4 - Sentiment and Binary Naive Bayes

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



# Let's do a worked sentiment 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               |

# A worked sentiment example with add-1 smoothing



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

3. Likelihoods from training:

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

$$\begin{aligned} P(\text{"predictable"}|-) &= \frac{1+1}{14+20} & P(\text{"predictable"}|+) &= \frac{0+1}{9+20} \\ P(\text{"no"}|-) &= \frac{1+1}{14+20} & P(\text{"no"}|+) &= \frac{0+1}{9+20} \\ P(\text{"fun"}|-) &= \frac{0+1}{14+20} & P(\text{"fun"}|+) &= \frac{1+1}{9+20} \end{aligned}$$

1. Prior from training:

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

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

2. Drop "with"

4. Scoring the test set:

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

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

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 multinomial naive bayes, or binary NB

- Clip our word counts at 1
- Note: this is different than Bernoulli naive bayes; see the textbook at the end of the chapter.

- 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  $P(w_k | c_j)$  terms
  - Remove duplicates containing all  $docs_j$
  - For each word type  $w_k$  in  $Vocabulary$ 
    - Retain only a single instance of  $w_k$

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

## Binary Multinomial Naive Bayes on a test document $d$



First remove all duplicate words from  $d$

Then compute NB using the same equation:

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

## 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

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

# SIT330-770: Natural Language Processing

Week 5.5 - More on Sentiment Classification

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



- I really like this movie
- I really **don't** like this movie

Negation changes the meaning of "like" to negative.

Negation can also change negative to positive-ish

- **Don't** dismiss this film
- **Doesn't** let us get bored

# Sentiment Classification: Dealing with Negation



Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

Simple baseline method:

Add NOT\_ to every word between negation and following punctuation:

didn't like this movie , but I



didn't NOT\_like NOT\_this NOT\_movie but I

- Sometimes we don't have enough labeled training data
- In that case, we can make use of pre-built word lists
- Called **lexicons**
- There are various publically available lexicons

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proc. of HLT-EMNLP-2005.

Riloff and Wiebe (2003). Learning extraction patterns for subjective expressions. EMNLP-2003.

Home page: [https://mpqa.cs.pitt.edu/lexicons/subj\\_lexicon/](https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/)

6885 words from 8221 lemmas, annotated for intensity (strong/weak)

- 2718 positive
- 4912 negative

+ : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*

- : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

Philip J. Stone, Dexter C Dunphy, Marshall S. Smith, Daniel M. Ogilvie. 1966. The General Inquirer: A Computer Approach to Content Analysis. MIT Press

- Home page: <http://www.wjh.harvard.edu/~inquirer>
- List of Categories: <http://www.wjh.harvard.edu/~inquirer/homecat.htm>
- Spreadsheet: <http://www.wjh.harvard.edu/~inquirer/inquirerbasic.xls>
- Categories:
  - Positiv (1915 words) and Negativ (2291 words)
  - Strong vs Weak, Active vs Passive, Overstated versus Understated
  - Pleasure, Pain, Virtue, Vice, Motivation, Cognitive Orientation, etc
- Free for Research Use

Add a feature that gets a count whenever a word from the lexicon occurs

- E.g., a feature called "**this word occurs in the positive lexicon**" or "**this word occurs in the negative lexicon**"

Now all positive words (*good, great, beautiful, wonderful*) or negative words count for that feature.

Using 1-2 features isn't as good as using all the words.

- But when training data is sparse or not representative of the test set, dense lexicon features can help

- SpamAssassin Features:
  - Mentions millions of (dollar) ((dollar) NN,NNN,NNN.NN)
  - From: starts with many numbers
  - Subject is all capitals
  - HTML has a low ratio of text to image area
  - "One hundred percent guaranteed"
  - Claims you can be removed from the list

- Determining what language a piece of text is written in.

Features based on character n-grams do very well

- Important to train on lots of varieties of each language
  - (e.g., American English varieties like African-American English, or English varieties around the world like Indian English)

# Summary: Naive Bayes is Not So Naive



- 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
  - **But we will see other classifiers that give better accuracy**

# SIT330-770: Natural Language Processing

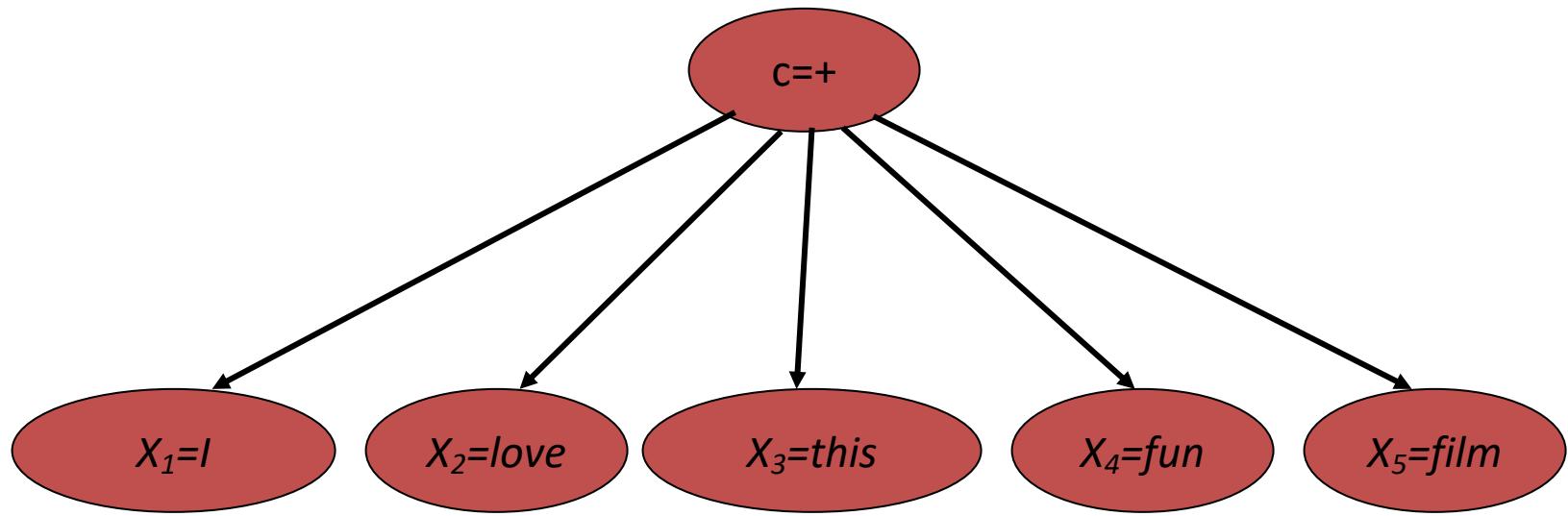
Week 5.6 - Naïve Bayes: Relationship to Language Modeling

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



# Generative Model for Multinomial Naïve Bayes



- Naïve bayes classifiers can use any sort of feature
  - URL, email address, dictionaries, network features
- But if, as in the previous slides
  - We use **only** word features
  - we use **all** of the words in the text (not a subset)
- Then
  - Naive bayes has an important similarity to language modeling.

- Assigning each word:  $P(\text{word} \mid c)$
- Assigning each sentence:  $P(s \mid c) = \prod P(\text{word} \mid c)$

Class pos

|      |      |  |     |      |      |      |                                    |
|------|------|--|-----|------|------|------|------------------------------------|
| 0.1  | I    |  | I   | love | this | fun  | film                               |
| 0.1  | love |  | 0.1 | 0.1  | .05  | 0.01 | 0.1                                |
| 0.01 | this |  |     |      |      |      |                                    |
| 0.05 | fun  |  |     |      |      |      |                                    |
| 0.1  | film |  |     |      |      |      |                                    |
|      | ...  |  |     |      |      |      |                                    |
|      |      |  |     |      |      |      | $P(s \mid \text{pos}) = 0.0000005$ |

- Which class assigns the higher probability to s?

| Model pos |      | Model neg |      |          |             |             |            |             |
|-----------|------|-----------|------|----------|-------------|-------------|------------|-------------|
| 0.1       | I    | 0.2       | I    | <u>I</u> | <u>love</u> | <u>this</u> | <u>fun</u> | <u>film</u> |
| 0.1       | love | 0.001     | love | 0.1      | 0.1         | 0.01        | 0.05       | 0.1         |
| 0.01      | this | 0.01      | this | 0.2      | 0.001       | 0.01        | 0.005      | 0.1         |
| 0.05      | fun  | 0.005     | fun  |          |             |             |            |             |
| 0.1       | film | 0.1       | film |          |             |             |            |             |

$P(s|pos) > P(s|neg)$

# SIT330-770: Natural Language Processing

Week 5.7 – Evaluating a Sentiment Classifier

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



- Let's consider just binary text classification tasks
- Imagine you're the CEO of Delicious Pie Company
- You want to know what people are saying about your pies
- So you build a "Delicious Pie" tweet detector
  - Positive class: tweets about Delicious Pie Co
  - Negative class: all other tweets

# The 2-by-2 confusion matrix



|                             |                 | <i>gold standard labels</i>        |   |
|-----------------------------|-----------------|------------------------------------|---|
|                             |                 | gold positive                      | gold negative                                 |
| <i>system output labels</i> | system positive | <b>true positive</b>               | <b>false positive</b>                         |
|                             | system negative | <b>false negative</b>              | <b>true negative</b>                          |
|                             |                 | <b>recall</b> = $\frac{tp}{tp+fn}$ | <b>precision</b> = $\frac{tp}{tp+fp}$         |
|                             |                 |                                    | <b>accuracy</b> = $\frac{tp+tn}{tp+fp+tn+fn}$ |

- Why don't we use **accuracy** as our metric?
- Imagine we saw 1 million tweets
  - 100 of them talked about Delicious Pie Co.
  - 999,900 talked about something else
- We could build a dumb classifier that just labels every tweet "not about pie"
  - It would get 99.99% accuracy!!! Wow!!!!
  - But useless! Doesn't return the comments we are looking for!
  - That's why we use **precision** and **recall** instead

- % of items the system detected (i.e., items the system labeled as positive) that are in fact positive (according to the human gold labels)

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- % of items actually present in the input that were correctly identified by the system.

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- Our dumb pie-classifier
  - Just label nothing as "about pie"

Accuracy=99.99%

but

Recall = 0

- (it doesn't get any of the 100 Pie tweets)

Precision and recall, unlike accuracy, emphasize true positives:

- finding the things that we are supposed to be looking for.

## A combined measure: F



- F measure: a single number that combines P and R:

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2P + R}$$

- We almost always use balanced  $F_1$  (i.e.,  $\beta = 1$ )

$$F_1 = \frac{2PR}{P + R}$$

Training set

Development Test Set

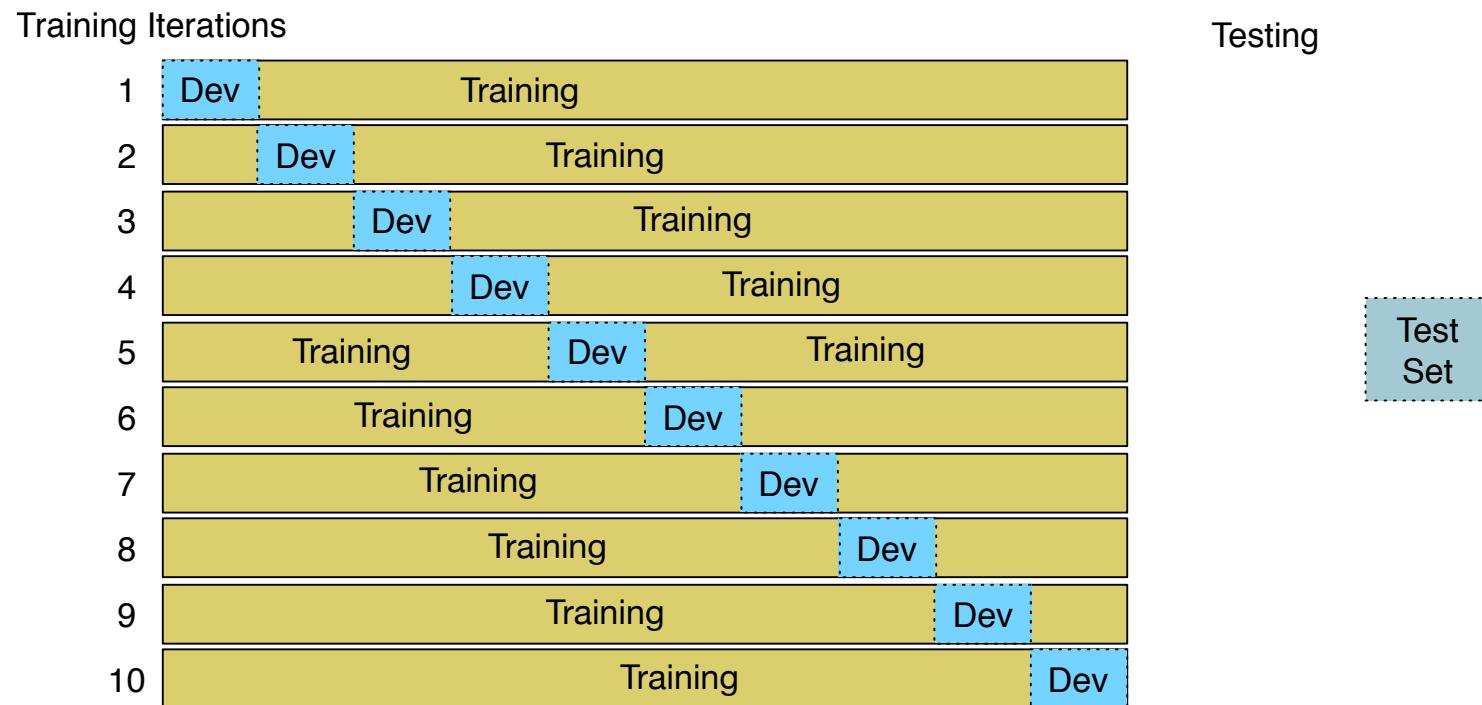
Test Set

- Train on training set, tune on devset, report on testset
  - This avoids overfitting ('tuning to the test set')
  - More conservative estimate of performance
  - But paradox: want as much data as possible for training, and as much for dev; how to split?

# Cross-validation: multiple splits



- Pool results over splits, Compute pooled dev performance



# SIT330-770: Natural Language Processing

Week 5.8 – Evaluation with more than two classes

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



# Confusion Matrix for 3-class classification



|                      |        | <i>gold labels</i> |        |      |
|----------------------|--------|--------------------|--------|------|
|                      |        | urgent             | normal | spam |
| <i>system output</i> | urgent | 8                  | 10     | 1    |
|                      | normal | 5                  | 60     | 50   |
|                      | spam   | 3                  | 30     | 200  |

**precision<sub>u</sub>**=  $\frac{8}{8+10+1}$

**precision<sub>n</sub>**=  $\frac{60}{5+60+50}$

**precision<sub>s</sub>**=  $\frac{200}{3+30+200}$

**recall<sub>u</sub>**=  $\frac{8}{8+5+3}$

**recall<sub>n</sub>**=  $\frac{60}{10+60+30}$

**recall<sub>s</sub>**=  $\frac{200}{1+50+200}$

# How to combine P/R from 3 classes to get one metric



- Macroaveraging:
  - compute the performance for each class, and then average over classes
- Microaveraging:
  - collect decisions for all classes into one confusion matrix
  - compute precision and recall from that table.

# Macroaveraging and Microaveraging



**Class 1: Urgent**

|        |        | true   | true |
|--------|--------|--------|------|
|        |        | urgent | not  |
| system | urgent | 8      | 11   |
|        | not    | 8      | 340  |

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

**Class 2: Normal**

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

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

**Class 3: Spam**

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

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

**Pooled**

|        |     | true | true |
|--------|-----|------|------|
|        |     | yes  | no   |
| system | yes | 268  | 99   |
|        | no  | 99   | 635  |

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

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

# SIT330-770: Natural Language Processing

Week 5.9 – Statistical Significance Testing

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



# How do we know if one classifier is better than another?



- Given:
  - Classifier A and B
  - Metric M:  $M(A, x)$  is the performance of A on testset x
  - $\delta(x)$ : the performance difference between A, B on x:
    - $\delta(x) = M(A, x) - M(B, x)$
  - We want to know if  $\delta(x) > 0$ , meaning A is better than B
  - $\delta(x)$  is called the **effect size**
  - Suppose we look and see that  $\delta(x)$  is positive. Are we done?
  - No! This might be just an accident of this one test set, or circumstance of the experiment. Instead:

- Consider two hypotheses:
  - Null hypothesis: A isn't better than B  $H_0 : \delta(x) \leq 0$
  - A is better than B  $H_1 : \delta(x) > 0$
- We want to rule out  $H_0$
- We create a random variable X ranging over test sets
- And ask, how likely, if  $H_0$  is true, is it that among these test sets we would see the  $\delta(x)$  we did see?
  - Formalized as the p-value:

$$P(\delta(X) \geq \delta(x) | H_0 \text{ is true})$$

$$P(\delta(X) \geq \delta(x) | H_0 \text{ is true})$$

- In our example, this p-value is the probability that we would see  $\delta(x)$  assuming  $H_0$  ( $=A$  is not better than  $B$ ).
  - If  $H_0$  is true but  $\delta(x)$  is huge, that is surprising! Very low probability!
- A very small p-value means that the difference we observed is very unlikely under the null hypothesis, and we can reject the null hypothesis
  - Very small: .05 or .01
  - A result(e.g., “ $A$  is better than  $B$ ”) is **statistically significant** if the  $\delta$  we saw has a probability that is below the threshold and we therefore reject this null hypothesis.

- How do we compute this probability?
- In NLP, we don't tend to use parametric tests (like t-tests)
- Instead, we use non-parametric tests based on sampling: artificially creating many versions of the setup.
- For example, suppose we had created zillions of testsets  $x'$ .
  - Now we measure the value of  $\delta(x')$  on each test set
  - That gives us a distribution
  - Now set a threshold (say .01).
  - So if we see that in 99% of the test sets  $\delta(x) > \delta(x')$ 
    - We conclude that our original test set delta was a real delta and not an artifact.

- Two common approaches:
  - approximate randomization
  - bootstrap test
- Paired tests:
  - Comparing two sets of observations in which each observation in one set can be paired with an observation in another.
  - For example, when looking at systems A and B **on the same test set**, we can compare the performance of system A and B on each same observation  $x_i$

# SIT330-770: Natural Language Processing

Week 5.10 – The Paired Bootstrap Test

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



Efron and Tibshirani, 1993

Can apply to any metric (accuracy, precision, recall,  $F_1$ , etc).

**Bootstrap** means to repeatedly draw large numbers of smaller samples with replacement (called **bootstrap samples**) from an original larger sample.

Consider a baby text classification example with a test set  $x$  of 10 documents, using accuracy as metric.

Suppose these are the results of systems A and B on  $x$ , with 4 outcomes (A & B both right, A & B both wrong, A right/B wrong, A wrong/B right):

either A+B both correct, or

|     | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | A%  | B%  | $\delta()$ |
|-----|----|----|----|----|----|----|----|----|----|----|-----|-----|------------|
| $x$ | AB | .70 | .50 | .20        |

- Now we create, many, say,  $b=10,000$  virtual test sets  $x(i)$ , each of size  $n = 10$ .
- To make each  $x(i)$ , we randomly select a cell from row  $x$ , with replacement,  
10 times:

|           | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | A%  | B%  | $\delta()$ |
|-----------|----|----|----|----|----|----|----|----|----|----|-----|-----|------------|
| $x$       | AB | .70 | .50 | .20        |
| $x^{(1)}$ | AB | .60 | .60 | .00        |
| $x^{(2)}$ | AB | .60 | .70 | -.10       |
| ...       |    |    |    |    |    |    |    |    |    |    |     |     |            |
| $x^{(b)}$ |    |    |    |    |    |    |    |    |    |    |     |     |            |

- Now we have a distribution! We can check how often A has an **accidental** advantage, to see if the original  $\delta(x)$  we saw was very common.
- Now assuming  $H_0$ , that means normally we expect  $\delta(x')=0$
- So we just count how many times the  $\delta(x')$  we found exceeds the expected value by  $\delta(x)$  or more:

$$\text{p-value}(x) = \sum_{i=1}^b \mathbb{1} \left( \delta(x^{(i)}) - \delta(x) \geq 0 \right)$$

- Alas, it's slightly more complicated.
- We didn't draw these samples from a distribution with 0 mean; we created them from the original test set  $x$ , which happens to be biased (by .20) in favor of A.
- So to measure how surprising is our observed  $\delta(x)$ , we actually compute the p-value by counting how often  $\delta(x')$  exceeds the expected value of  $\delta(x)$  by  $\delta(x)$  or more:

$$\begin{aligned} \text{p-value}(x) &= \sum_{i=1}^b \mathbb{1} \left( \delta(x^{(i)}) - \delta(x) \geq \delta(x) \right) \\ &= \sum_{i=1}^b \mathbb{1} \left( \delta(x^{(i)}) \geq 2\delta(x) \right) \end{aligned}$$

- Suppose:
  - We have 10,000 test sets  $x(i)$  and a threshold of .01
  - And in only 47 of the test sets do we find that  $\delta(x(i)) \geq 2\delta(x)$
  - The resulting p-value is .0047
  - This is smaller than .01, indicating  $\delta(x)$  is indeed sufficiently surprising
  - And we reject the null hypothesis and conclude  $A$  is better than  $B$ .

After Berg-Kirkpatrick et al (2012)

**function** BOOTSTRAP(test set  $x$ , num of samples  $b$ ) **returns**  $p\text{-value}(x)$

Calculate  $\delta(x)$  # how much better does algorithm A do than B on  $x$

$s = 0$

**for**  $i = 1$  **to**  $b$  **do**

**for**  $j = 1$  **to**  $n$  **do** # Draw a bootstrap sample  $x^{(i)}$  of size  $n$

        Select a member of  $x$  at random and add it to  $x^{(i)}$

    Calculate  $\delta(x^{(i)})$  # how much better does algorithm A do than B on  $x^{(i)}$

$s \leftarrow s + 1$  **if**  $\delta(x^{(i)}) > 2\delta(x)$

$p\text{-value}(x) \approx \frac{s}{b}$  # on what % of the  $b$  samples did algorithm A beat expectations?

**return**  $p\text{-value}(x)$  # if very few did, our observed  $\delta$  is probably not accidental

# SIT330-770: Natural Language Processing

Week 5.10 – Text Classification: Practical Issues

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



- Gee, I'm building a text classifier for real, now!
- What should I do

If (wheat or grain) and not (whole or bread) then

Categorize as grain

- Need careful crafting
  - Human tuning on development data
  - Time-consuming: 2 days per class

- Use Naïve Bayes
  - Naïve Bayes is a “high-bias” algorithm ([Ng and Jordan 2002 NIPS](#))
- Get more labeled data
  - Find clever ways to get humans to label data for you
- Try semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, ...

# A reasonable amount of data?



- Perfect for all the clever classifiers
  - SVM
  - Regularized Logistic Regression
- You can even use user-interpretable decision trees
  - Users like to hack
  - Management likes quick fixes

## A huge amount of data?

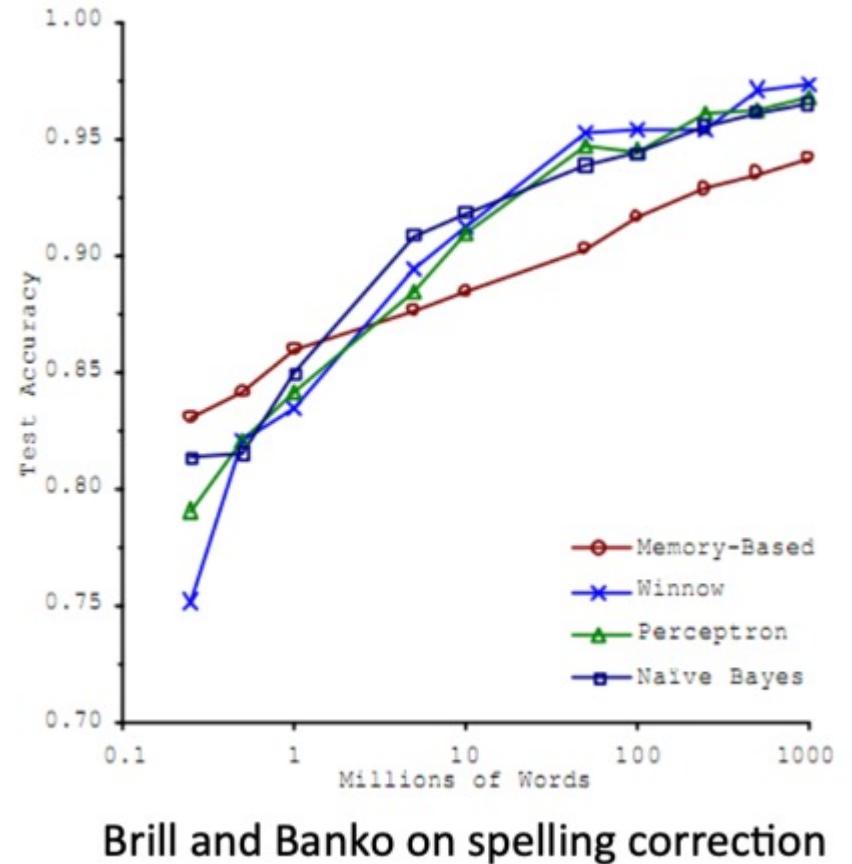


- Can achieve high accuracy!
- At a cost:
  - SVMs (train 1me) or kNN (test 1me) can be too slow
  - Regularized logistic regression can be somewhat better
- So Naïve Bayes can come back into its own again!

# Accuracy as a function of data size



- With enough data
  - Classifier may not matter



## Real-world systems generally combine:

- Automatic classification
- Manual review of uncertain/difficult/"new" cases



- Multiplying lots of probabilities can result in floating-point underflow.
- Since  $\log(xy) = \log(x) + \log(y)$ 
  - Better to sum logs of probabilities instead of multiplying probabilities.
- Class with highest un-normalized log probability score is still most probable.

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

- Model is now just max of sum of weights

- Domain-specific features and weights: *very important* in real performance
- Sometimes need to collapse terms:
  - Part numbers, chemical formulas, ...
  - But stemming generally doesn't help
- Upweighting: Counting a word as if it occurred twice:
  - title words ([Cohen & Singer 1996](#))
  - first sentence of each paragraph ([Murata, 1999](#))
  - In sentences that contain title words ([Ko et'al, 2002](#))

# SIT330-770: Natural Language Processing

Week 5.11 – Avoiding Harms in Classification

Dr. Mohamed Reda Bouadjenek

School of Information Technology,  
Faculty of Sci Eng & Built Env



- Kiritchenko and Mohammad (2018) found that most sentiment classifiers assign lower sentiment and more negative emotion to sentences with African American names in them.
- This perpetuates negative stereotypes that associate African Americans with negative emotions

- Toxicity detection is the task of detecting hate speech, abuse, harassment, or other kinds of toxic language
- But some toxicity classifiers incorrectly flag as being toxic sentences that are non-toxic but simply mention identities like blind people, women, or gay people.
- This could lead to censorship of discussion about these groups.

## What causes these harms?



- Can be caused by:
  - Problems in the training data; machine learning systems are known to amplify the biases in their training data.
  - Problems in the human labels
  - Problems in the resources used (like lexicons)
  - Problems in model architecture (like what the model is trained to optimized)
- Mitigation of these harms is an open research area
- Meanwhile: **model cards**

(Mitchell et al., 2019)

- For each algorithm you release, document:
  - training algorithms and parameters
  - training data sources, motivation, and preprocessing
  - evaluation data sources, motivation, and preprocessing
  - intended use and users
  - model performance across different demographic or other groups and environmental situations