UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Natural Language Processing

Text Classification: Sentiment Analysis

LESSON 9

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

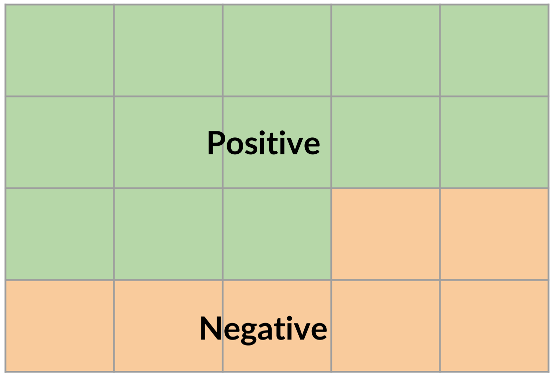
Sentiment Analysis

A probabilistic formulation: Towards Naïve Bayes

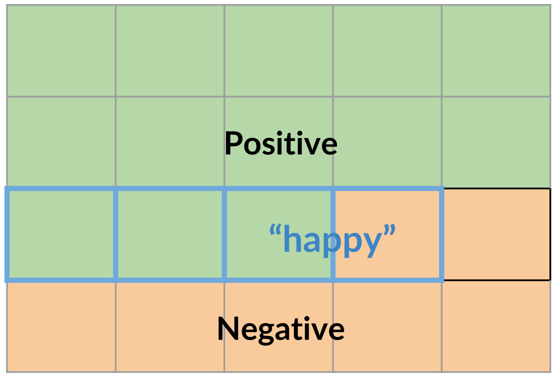
A probabilistic formulation of Sentiment Analysis

- Now, we turn to a probabilistic formulation of Sentiment Analysis
  - Based on Bayes' rule
- Suppose an extensive corpus of tweets that can be categorized as either positive or negative sentiment, but not both

Corpus of tweets



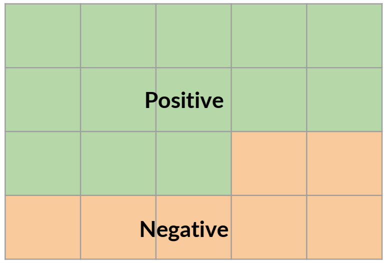
Tweets containing the word "happy"



Probabilities

- Define the event A as a tweet being labeled positive
  - the probability of event A is calculated as the ratio between the counts of positive tweets in the corpus divided by the total number of tweets in the corpus

Corpus of tweets



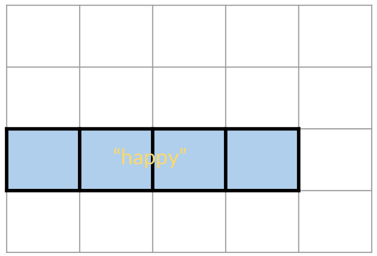
A → Positive tweet

$P(A) = N_{pos} / N = 13 / 20 = 0.65$

$P(Negative) = 1 - P(Positive) = 0.35$

- Let's define Event B in a similar way by counting tweets containing the word happy

Tweets containing the word "happy"



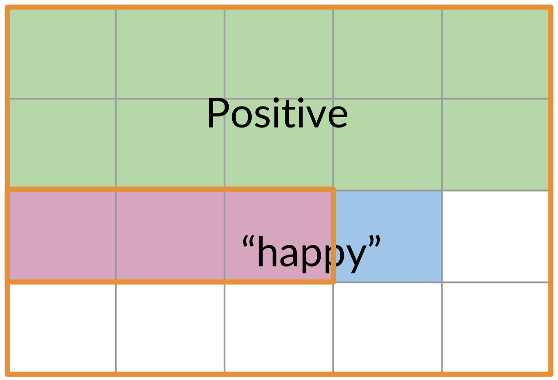
B → tweet contains "happy"

$P(B) = P(happy) = N_{happy} / N$

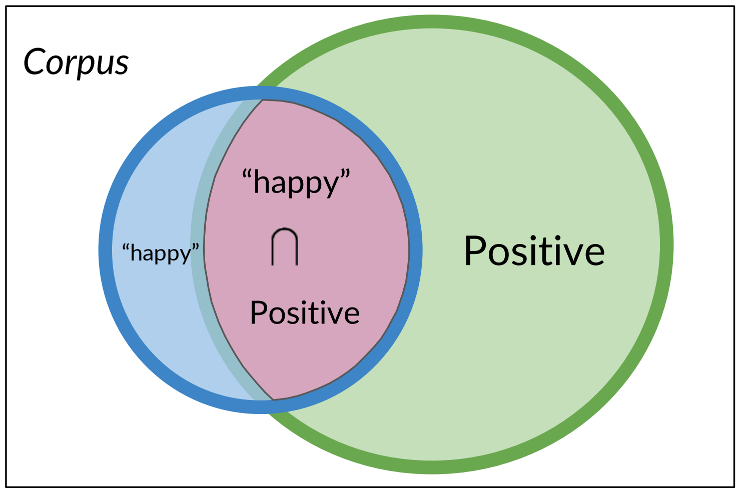
$P(B) = 4 / 20 = 0.2$

Probability of the intersection

- The probability that a tweet is labeled positive and contains the word happy is the ratio of the area of the intersection divided by the area of the entire corpus

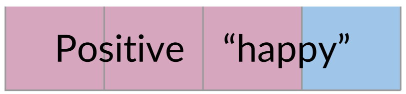


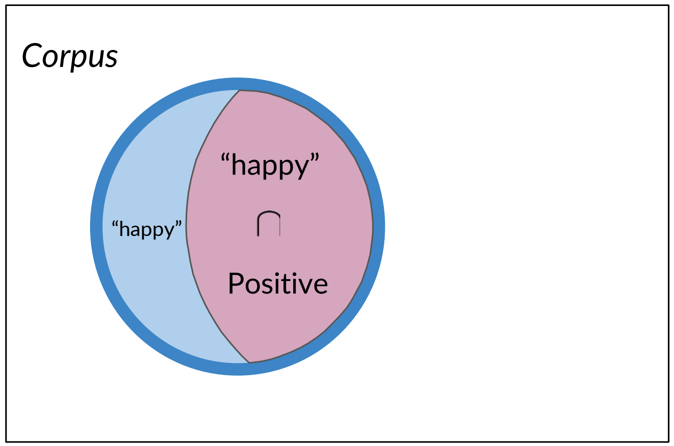
$$P(A \cap B) = P(A, B) = \frac{3}{20} = 0.15$$



Conditional probabilities

- Consider only tweets that contain the word happy
- the probability that a tweet is positive, given that it contains the word happy, is
  - the number of tweets that are positive and also contain the word happy, divided by the number that contain the word happy



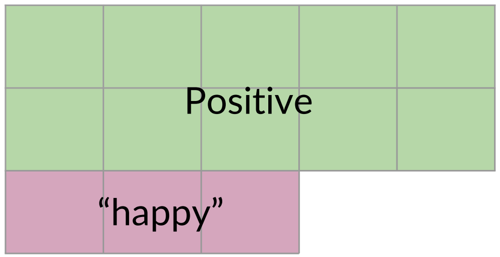


P(A | B) = P(Positive | "happy")

P(A | B) = 3 / 4 = 0.75

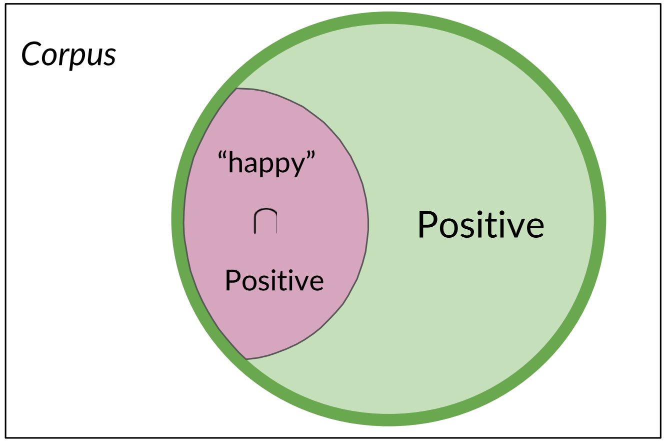
Conditional probabilities

- The same case for positive tweets



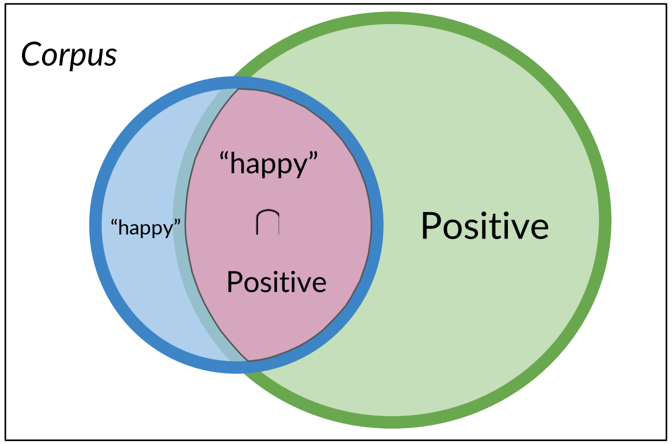
P(B | A) = P("happy"| Positive)

P(B | A) = 3 / 13 = 0.231



Conditional probabilities

- Conditional probabilities help reduce the sample search space
- For example, given a specific event already happened, i.e., we know the word is happy, one would only search in the blue circle below



$$P(\text{Positive} | \text{“happy”}) = \frac{P(\text{Positive} \cap \text{“happy”})}{P(\text{“happy”})}$$

Bayes' Rule

$$P(Positive|"happy") = \frac{P(Positive \cap "happy")}{P("happy")}$$

$$P("happy"|Positive) = \frac{P("happy" \cap Positive)}{P(Positive)}$$

$$P(Positive|"happy") = P("happy"|Positive) \times \frac{P(Positive)}{P("happy")}$$

- Let's recall the general Bayes rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

Naïve Bayes classifier

- $\hat{c} = \underset{c \in C}{\text{argmax}}\, P(c|d) = \underset{c \in C}{\text{argmax}}\, \frac{P(d|c)P(c)}{P(d)} = \underset{c \in C}{\text{argmax}}\, P(d|c)P(c)$

- Generative model
  - Defines how a document is generated
    - Sample a class with probability P(c), then
    - Words generated by sampling from P(d|c)

- In general, we represent a document as a set of features
  - $\hat{c} = \underset{c \in C}{\text{argmax}}\, P(f_1, f_2, \dots, f_n|c)P(c)$

Naïve Bayes Assumptions

- Naïve Bayes makes the independence assumption between features associated with each class
- Example1
  - *"It is sunny and hot in the Sahara desert"*
  - the words *sunny* and *hot* tend to depend on each other and are correlated to a certain extent with the word *desert*
- Example 2
  - *"It's always cold and snowy in ___"*
  - if you were to fill in the sentence above, the model will assign equal weight to the words *spring*, *summer*, *fall*, *winter*

spring?? summer? fall?? winter??

Naïve Bayes Assumptions formally

- Naïve Bayes assumption
  - $\boldsymbol{P(f_1, f_2, \dots, f_n|c) = P(f_1|c)P(f_2|c)\dots P(f_n|c)}$
- Naïve Bayes classifier
  - $\boldsymbol{C_{NB} = arg\max_{c \in C} P(c) \prod_f P(f|c)}$
- To apply NB to text, word positions need to be considered
  - Positions <- all word positions in the test document
    - $C_{NB} = arg\max_{c \in C} P(c) \prod_{i \in positions} P(w_i|c)$

Learning the Bayes Model

- Maximum likelihood estimates
  - Simply use the frequencies in the data

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

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

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

Naïve Bayes for Sentiment Analysis

- Determine the word counts for each occurrence of a word in the positive and negative corpora

Positive tweets

I am happy because I am learning NLP

I am happy

Negative tweets

I am sad, I am not learning NLP

I am sad

word	Pos	Neg
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	**13**	**13**

Naïve Bayes for Sentiment Analysis

- Compute the conditional probabilities of each word given the class

word	Pos	Neg
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	13	13

$$P(w_i|class)$$

word	Pos	Neg
I	0.24	0.24
am	0.24	0.24
happy	0.15	0.08
because	0.08	0
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.15
not	0.08	0.15

Naïve Bayes

- Once obtained the probabilities, the likelihood score can be computed
  - A score greater than 1 indicates that the class is positive, otherwise negative
  - Let's suppose to have a new tweet:

Tweet: I am happy today; I am learning.

$$\prod_{i=1}^{m} \frac{P(wi|Pos)}{P(wi|Neg)} = \frac{0.15}{0.08} = 1.875 > 1$$

$$\frac{0.24}{0.24} \times \frac{0.24}{0.24} \times \frac{0.15}{0.08} \times \frac{0.24}{0.24} \times \frac{0.24}{0.24} \times \frac{0.08}{0.08}$$

word	Pos	Neg
I	0.24	0.24
am	0.24	0.24
happy	0.15	0.08
because	0.08	0
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.15
not	0.08	0.15

- Naïve Bayes condition rule for binary classification

$$\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)}$$

Laplacian Smoothing

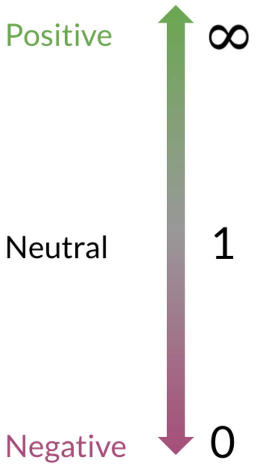
- We usually compute the probability of a word given a class as follows
  - $P(w_i | \text{class}) = \text{freq}(w_i, \text{class}) / N_{class}$ class ∈ { Positive, Negative }

- However, if a word does not appear in the training, then it automatically gets a probability of 0. To fix this, we add smoothing as follows
  - $P(w_i|\text{class}) = \text{freq}(w_i, \text{class}) + 1 / (N_{class} + V)$

- N_{class}: frequency of all words in class
- V: number of unique words in vocabulary

Ratio of probabilities

- Words can have many shades of emotional meaning
- For sentiment classification, they're simplified into three categories: neutral, positive, and negative
- All can be identified by using their conditional probabilities



word	Pos	Neg	ratio
I	0.20	0.20	1
am	0.20	0.20	1
happy	0.14	0.10	1.4
because	0.10	0.10	1
learning	0.10	0.10	1
NLP	0.10	0.10	1
sad	0.10	0.15	0.6
not	0.10	0.15	0.6

$$\text{ratio}(w_i) = P(w_i \mid \text{Pos}) \,/\, P(w_i \mid \text{Neg}) \approx \text{freq}(wi,1) + 1/\ (\text{freq}(wi,0)+1)$$

Naïve Bayes' inference

- Naïve Bayes formula for binary classification
  - Class ∈ { Positive, Negative }
  - w_i, i=1,…,m words in a tweet

$$\underbrace{\frac{P(pos)}{P(neg)}}_{\text{prior ratio}} \underbrace{\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)}}_{\text{likelihood}} > 1$$

Log Likelihood

- Sentiments probability calculation requires multiplication of many numbers with values between 0 and 1
  - risk of numerical underflow (values to small)
- Trick: use a log of the score instead of the raw score

$$log\left(\frac{P(pos)}{P(neg)}\prod_{i=1}^{n}\frac{P(w_i|pos)}{P(w_i|neg)}\right) \Rightarrow log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} log\frac{P(w_i|pos)}{P(w_i|neg)}$$

log prior + log likelihood

$$\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$$

lambda score

$$\prod_{i=1}^{m}\frac{P(w_i|pos)}{P(w_i|neg)} > 1$$

0 — 1 — ∞

Negative — Neutral — Positive

$$\sum_{i=1}^{m} log\frac{P(w_i|pos)}{P(w_i|neg)} > 0$$

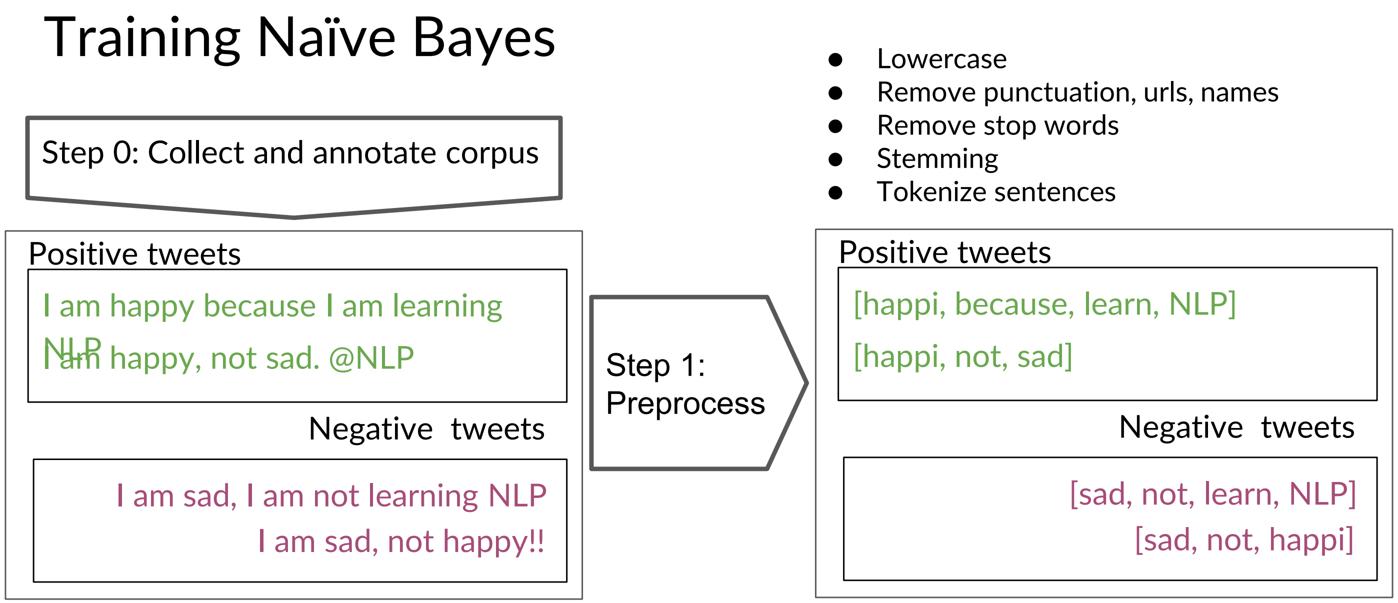
3.3 > 0

-∞ — 0 — ∞

Negative — Neutral — Positive

Training Naïve Bayes

- There is no gradient descent, just counting frequencies of words in the corpus
- Five steps for training a Naïve Bayes model



Training Naïve Bayes

- Start by computing the vocabulary for each word in class

Positive tweets

[happi, because, learn, NLP]

[happi, not, sad]

Negative tweets

[sad, not, learn, NLP]

[sad, not, happi]

Step 2: Word count

freq(w,class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Training Naïve Bayes

- Get the conditional probability (w. Laplacian smoothing)

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Step 3: $P(\mathrm{w}|\mathrm{class})$

$V = 6$

$$\frac{\mathrm{freq}(\mathrm{w}, \mathrm{class}) + 1}{N_{\mathrm{class}} + V}$$

$$\lambda(w) = log\frac{P(\mathrm{w}|\mathrm{pos})}{P(\mathrm{w}|\mathrm{neg})}$$

Step 4: Get lambda

word	Pos	Neg	λ
happy	0.23	0.15	0.43
because	0.15	0.07	0.6
learning	0.08	0.08	0
NLP	0.08	0.08	0
sad	0.08	0.17	-0.75
not	0.08	0.17	-0.75

Training Naïve Bayes

- Estimate the log prior
  - count the number of positive and negative tweets

Step 5: Get the log prior

D_{pos} = Number of positive tweets
D_{neg} = Number of negative tweets

$$\text{logprior} = log\frac{D_{pos}}{D_{neg}}$$

If dataset is balanced, $D_{pos} = D_{neg}$ and logprior = 0.

Training Naïve Bayes: Recap

1. Get or annotate a dataset with positive and negative tweets
2. Preprocess the tweets -> $[w_1, w_2, w_3, \ldots]$
3. Compute freq(w,class)
4. Get P(w|Pos) and P(w|Neg)
5. Get lambda(w)
6. Compute log prior = log(P(Pos)/P(Neg))

Unknown words

- What about unknown words
  - Appearing in test data
  - Not appearing in training data or vocabulary
- We ignore them
  - Removed 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 the training set
  - Call the top 10 or 50 words in the stop word list
  - Remove all stop words from both training and test sets
- But removing stop words doesn't usually help
  - In practice, most NB algorithms use all words and don't use stop word list

Testing Naïve Bayes

- Performance on unseen data -> X_{val} Y_{val}
- Predict using λ and log prior for each new tweet
- Accuracy

$$\frac{1}{m}\sum_{i=1}^{m}(pred_i == Yval_i)$$

- Words that not appear in $\lambda(m)$
  - treated as neutral words!

Evaluation

- 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

gold standard labels

		gold positive	gold negative	
system output labels	system positive	**true positive**	**false positive**	**precision** $= \frac{\text{tp}}{\text{tp+fp}}$
	system negative	**false negative**	**true negative**	
		recall $= \frac{\text{tp}}{\text{tp+fn}}$		**accuracy** $= \frac{\text{tp+tn}}{\text{tp+fp+tn+fn}}$

Evaluation: Accuracy

- 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

Evaluation: Precision

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

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

Evaluation: Recall

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

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

Why Precision and recall

- 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:

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

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

$$\mathrm{F}_1 = \frac{2PR}{P + R}$$

Naïve Bayes Assumptions

- Naïve Bayes is affected by the word frequencies in the corpus
- Example
  - On Twitter, there are usually more positive tweets than negative ones
  - However, some "clean" datasets you may find are artificially balanced to have to the same amount of positive and negative tweets
  - Just keep in mind, that in the real world, the data could be much noisier

Applications of Naïve Bayes

- There are many applications of naive Bayes including:
  - Author identification
  - Spam filtering
  - Information retrieval
  - Word disambiguation
  - This method is usually used as a simple baseline, and it is also fast

Applications of Naïve Bayes

Author identification:

P( | book)
/
P( | book)

Spam filtering:

$$\frac{P(\text{spam}|\text{email})}{P(\text{nonspam}|\text{email})}$$

Applications of Naïve Bayes

Information retrieval:

$$P(\mathrm{document_k}|\mathrm{query}) \propto \prod_{i=0}^{|query|} P(\mathrm{query_i}|\mathrm{document_k})$$

Retrieve document if $P(\mathrm{document_k}|\mathrm{query}) > \mathrm{threshold}$

Applications of Naïve Bayes

Word disambiguation:

$$\frac{P(\text{river}|\text{text})}{P(\text{money}|\text{text})}$$

Bank:  

Error Analysis

Source of errors in Naïve Bayes

- There are several mistakes that could cause you to misclassify an example or a tweet
  - Removing punctuation and stop words

Tweet: This is not good, because your attitude is not even close to being nice.

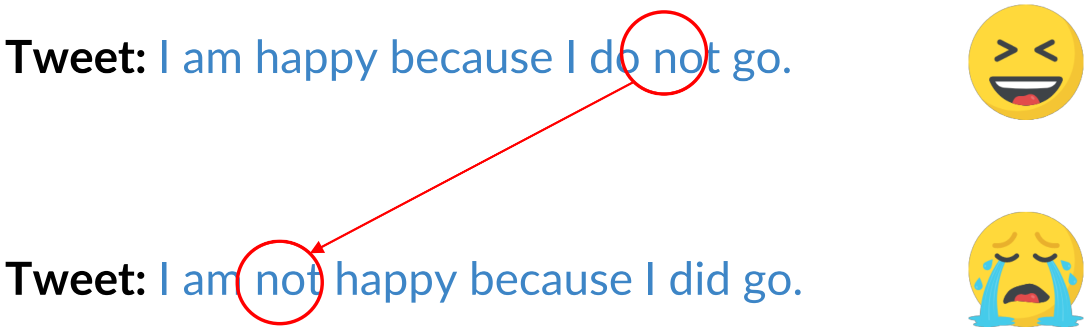
processed_tweet: [good, attitude, close, nice]

Tweet: My beloved grandmother :(

processed_tweet: [belov, grandmoth]

Source of errors in Naïve Bayes

- There are several mistakes that could cause you to misclassify an example or a tweet
  - Word order



Source of errors in Naïve Bayes

- There are several mistakes that could cause you to misclassify an example or a tweet
  - Adversarial attacks
    - Sarcasm, Irony and Euphemisms

Tweet: This is a ridiculously powerful movie. The plot was gripping and I cried right through until the ending!

processed_tweet: [ridicul, power, movi, plot, grip, cry, end]

Harms in Sentiment Classifiers

- 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

Harms in toxicity classification

- 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 discussions 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 optimize)
- Mitigation of these harms is an open research area
- Meanwhile: **model cards**

Model cards

- 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
- (Mitchell et al., 2019)

In Summary: 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 the results
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for the problem
- A good dependable baseline for text classification
  - But we know that other classifiers give better accuracies