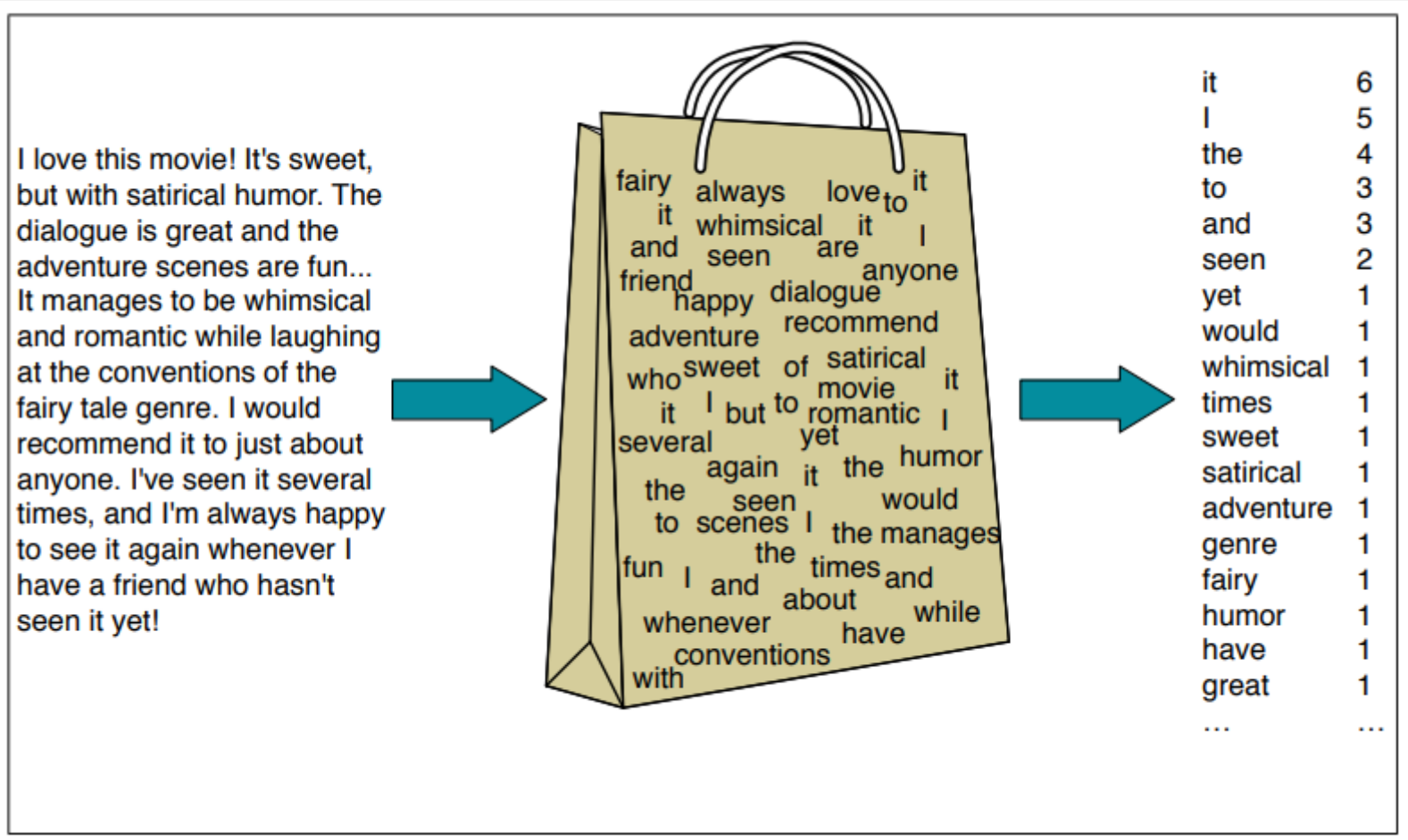
Intro to NLP
Sentiment Analysis with Naïve-Bayes

Gennaro S. Rodrigues, Ph.D.

Overview

1. Introduction to Bayes Rule
2. Probability and Log Likelihood
3. Naïve Bayes Algorithm
4. Applying Naïve Bayes

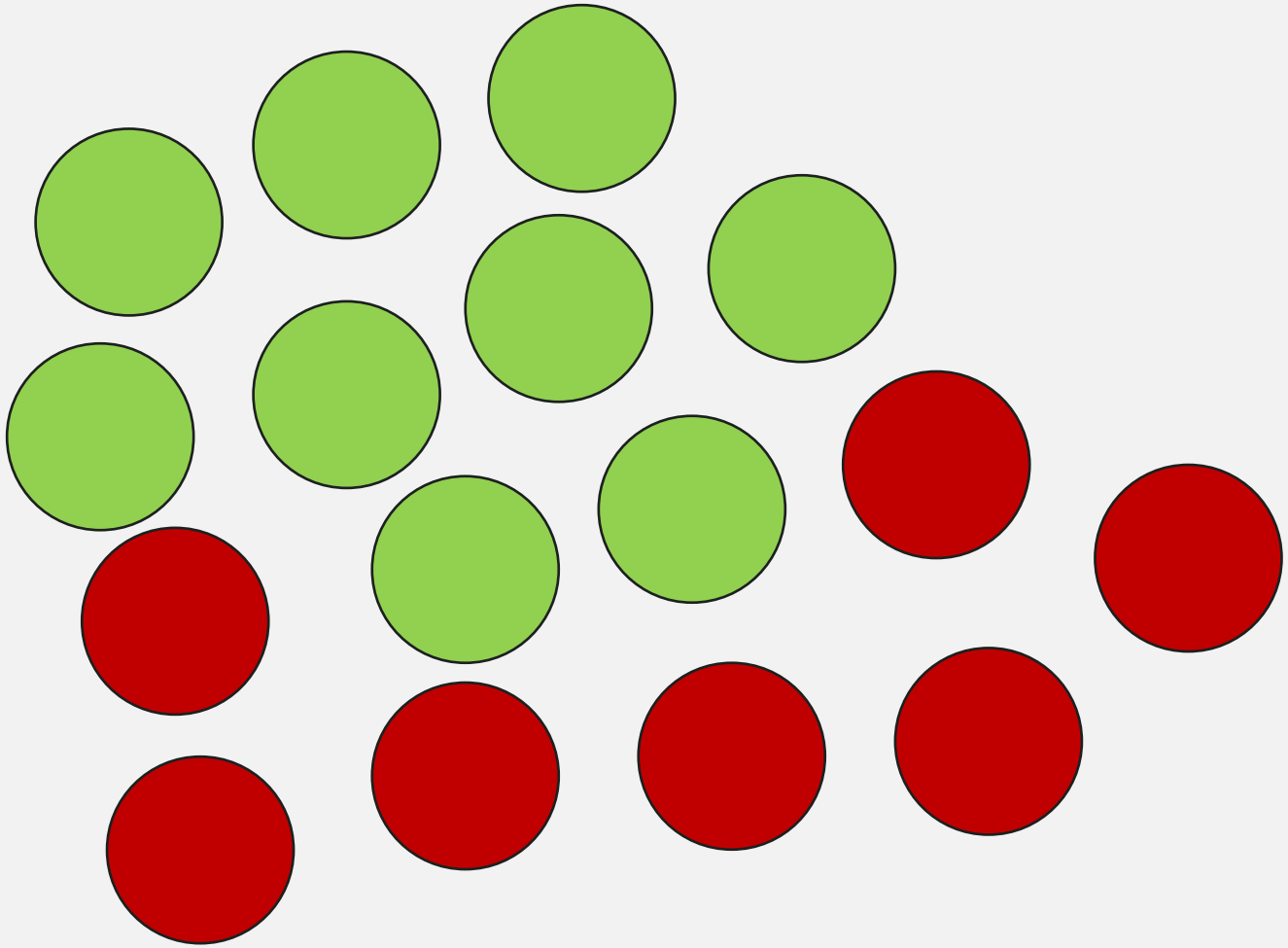
Naïve Bayes - BoW



(Source: Jurafsky et. Al., Speech and Language Processing [3rd ed. draft])

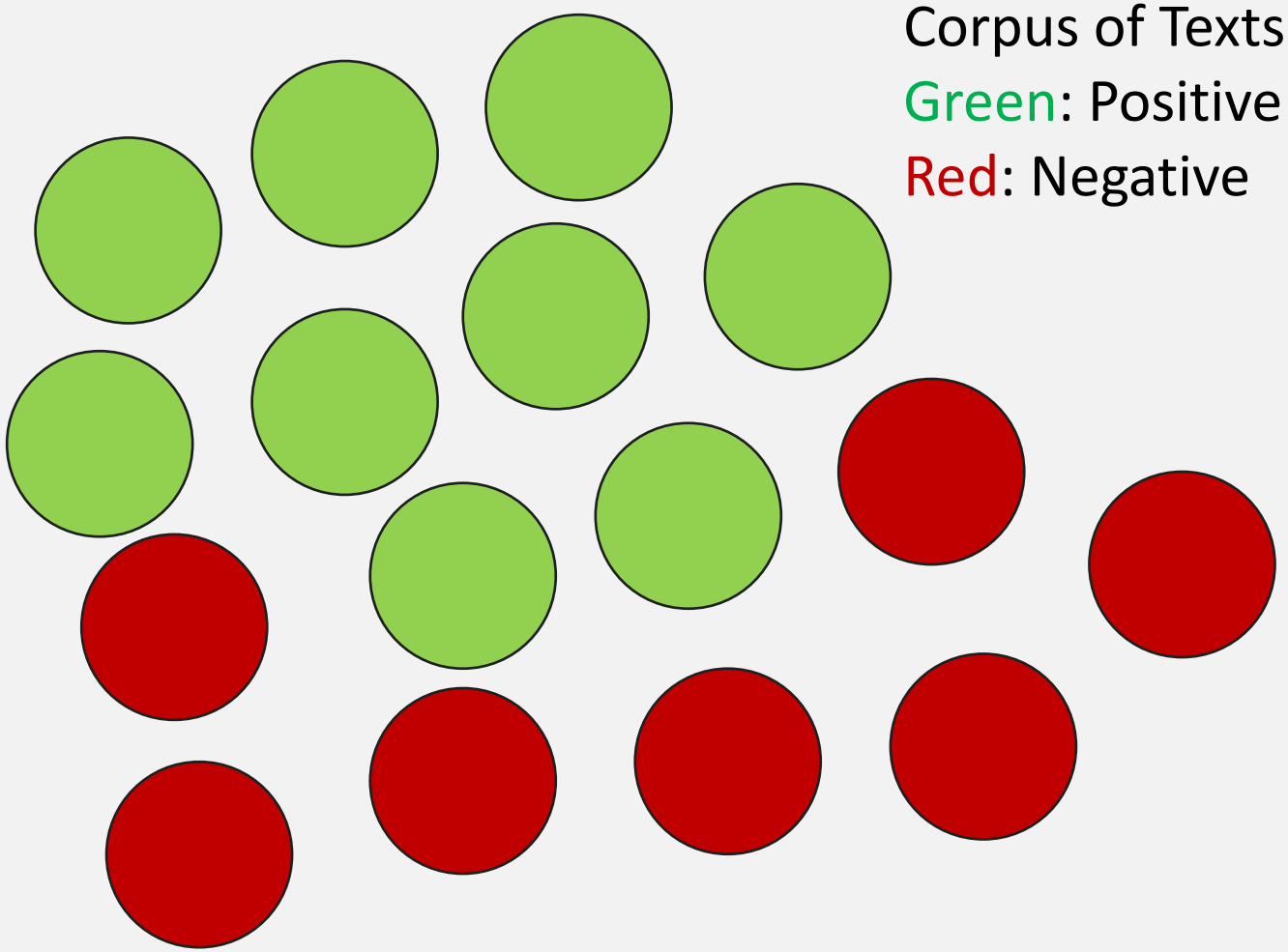
Introduction to Bayes Rule

Corpus of Texts
Green: Positive
Red: Negative



Introduction to Bayes Rule

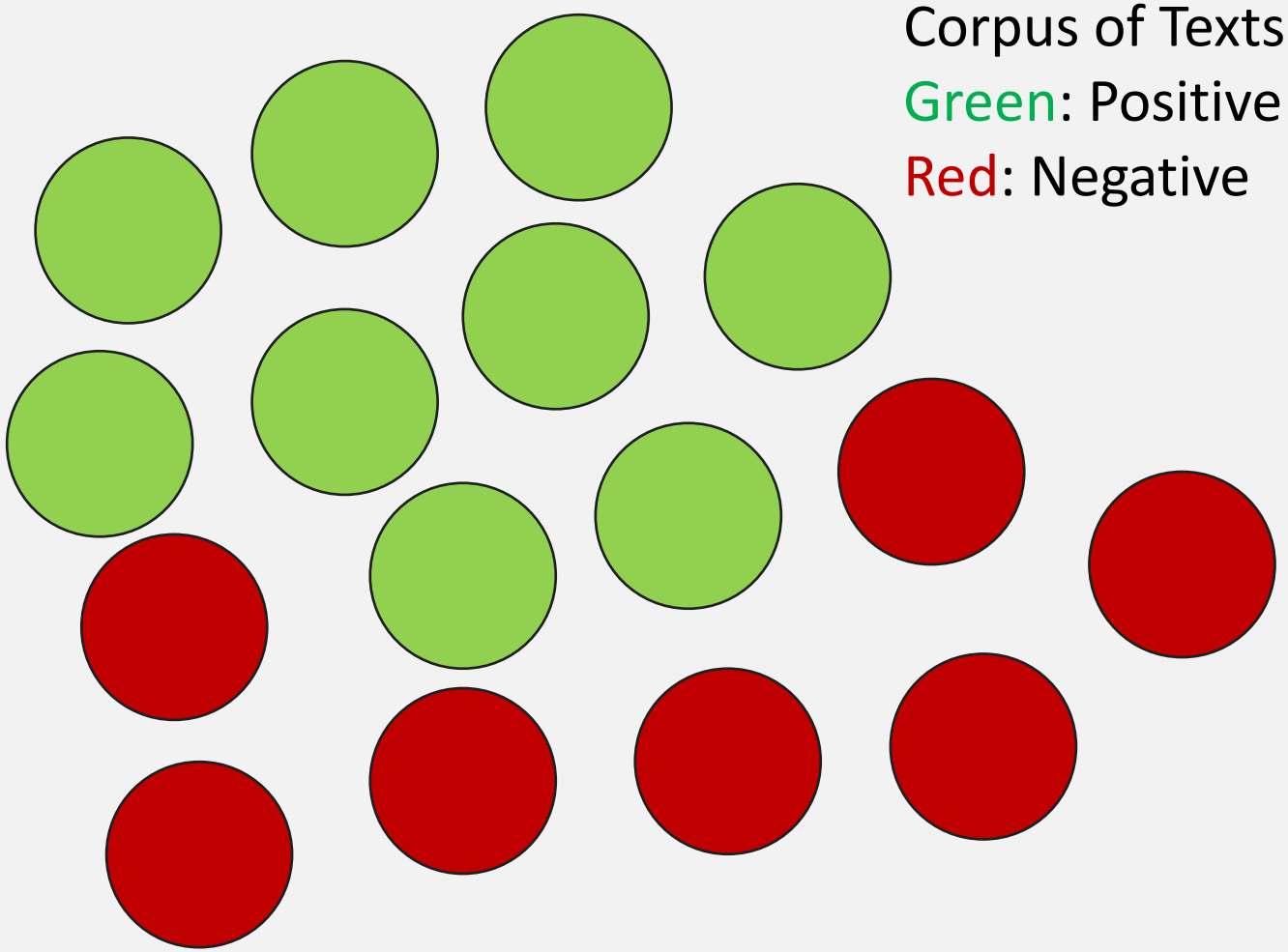
- Probability: frequency of occurrence.
- $P(Negative) = {}^{N_{Negative}}/_{N}$



Corpus of Texts
Green: Positive
Red: Negative

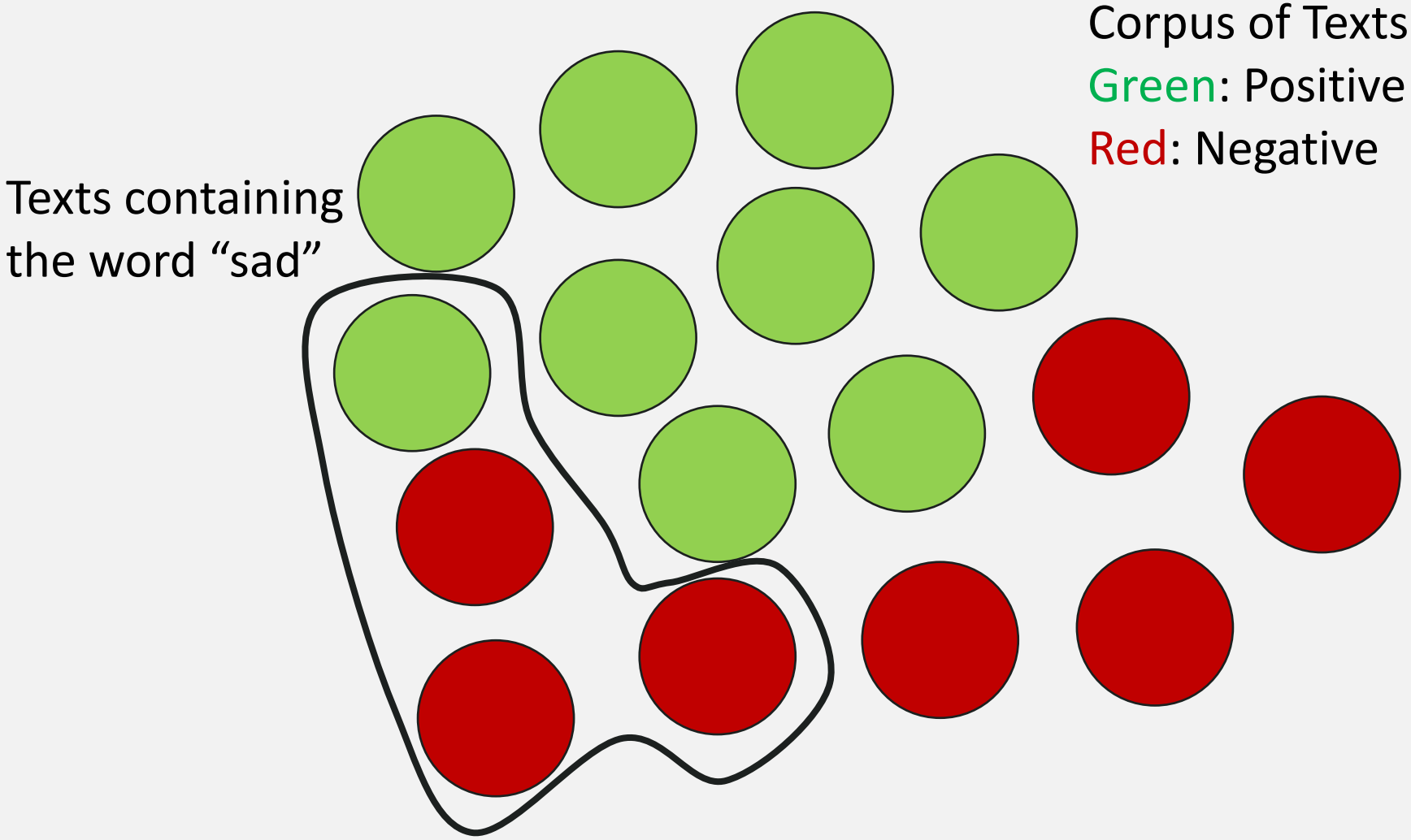
Introduction to Bayes Rule

- Probability: frequency of occurrence.
- $P(Negative) = {}^{N_{Negative}}/_{N} = {}^{7}/_{16} = 0.4375 = 43.75\%$
- $P(Positive) = 1 - P(Negative) = 56.25\%$



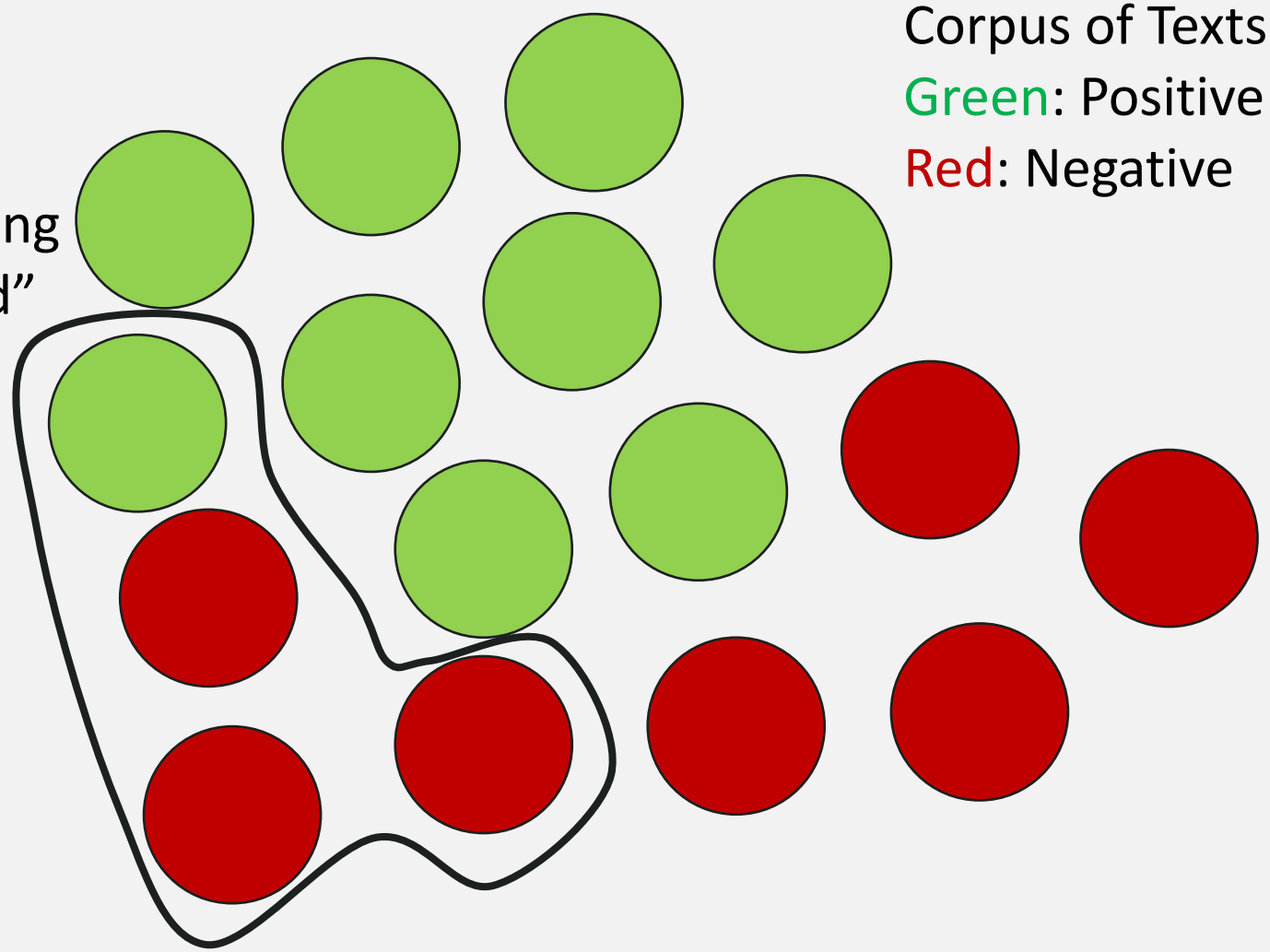
Corpus of Texts
Green: Positive
Red: Negative

Introduction to Bayes Rule

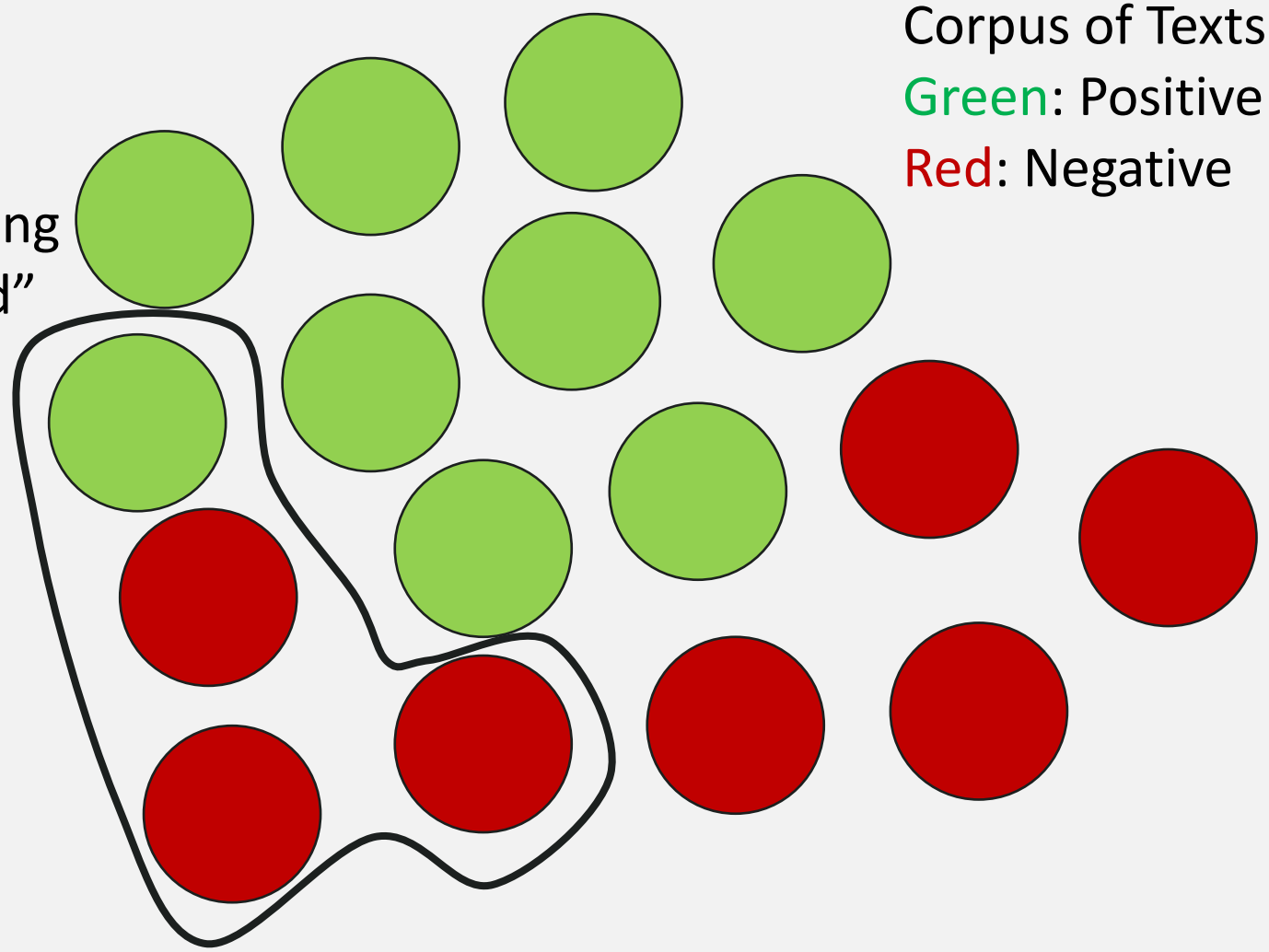


Introduction to Bayes Rule

- Probability: frequency of occurrence (word “sad”).
- $P(\text{"}sad\text{"}) = {}^{N_{sad}}/_{N} = {}^{4}/_{16} = 0.25$



Introduction to Bayes Rule

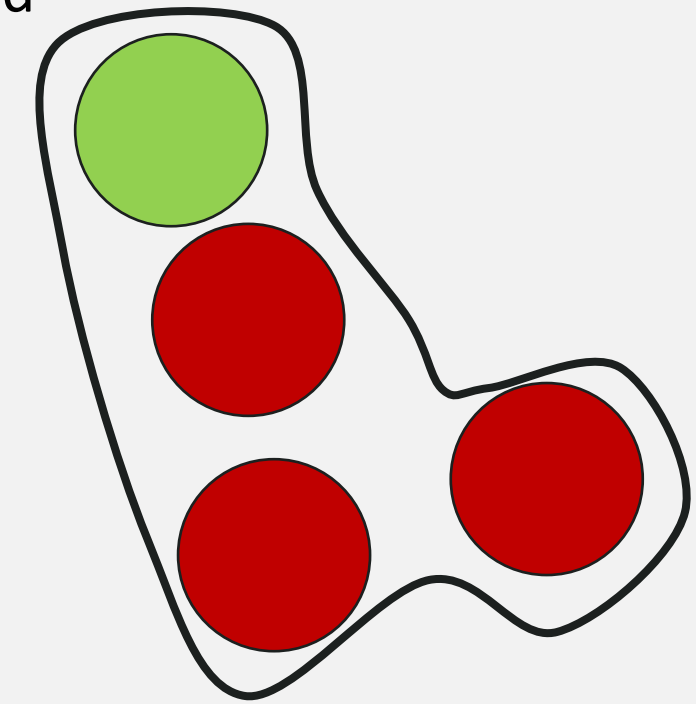


- Probability: frequency of occurrence (word “sad”).
- $P(\text{"}sad\text{"}) = {}^{N_{sad}}/_{N} = {}^{4}/_{16} = 0.25$
- $P(Negative \cap \text{"}sad\text{"}) = {}^{3}/_{16} = 0.1875 = P(Negative, \text{"}sad\text{"})$

Introduction to Bayes Rule

Corpus of Texts
Green: Positive
Red: Negative

Texts containing the word “sad”

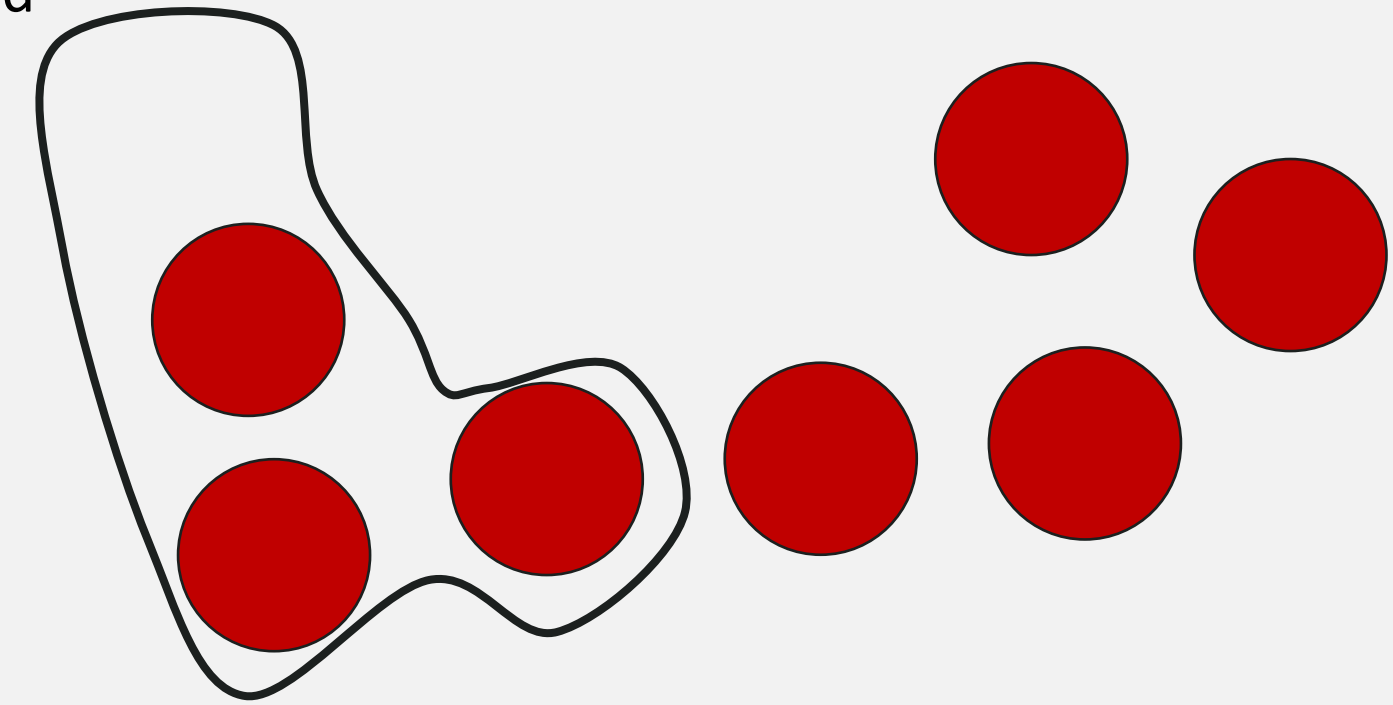


- Conditional Probability: probability of ***Negative*** provided that ***“sad”***.
- $P(Negative|"sad") = {}^{3}/_{4} = 0.75$

Introduction to Bayes Rule

Corpus of Texts
Green: Positive
Red: Negative

Texts containing the word “sad”



- Conditional Probability: probability of ***“sad”*** provided that ***Negative***.
- $P(\text{"}sad\text{"}|Negative) = {}^{3}/_{7} = 0.428$
- 42.8% of change for a text to contain the word “sad” if it is of negative sentiment.

Conditional Probability

- Probability of ***A*** provided that ***B*** *already happened.*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

$$P(B \cap A) = P(A \cap B)$$

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Conditional Probability

- Probability of ***A*** provided that ***B*** *already happened.*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

$$P(B \cap A) = P(A \cap B)$$

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Bayes Rule

Conditional Probability

- Probability of ***A*** provided that ***B*** *already happened.*

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Bayes Rule

- We can calculate the probability of A given B if we know the probability of B given A and the ratio of P(A) and P(B).

Naïve Bayes

- We can use the Bayes rule to classify sentiment on text.
- Supervised Machine Learning.
- Naïve: presupposes the features we use are independent. That is normally NOT the case. But it works well, nonetheless.

Naïve Bayes - Classification

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Bayes Rule

- We want to classify, for a given text, which class $c \in C$ has the *maximum posterior probability:*

$$\hat{c} = argmax_{c \in C} P(c|text)$$

Naïve Bayes - Classification

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Bayes Rule

$$\hat{c} = \ argmax_{c \in C} P(c|text)$$

Naïve Bayes - Classification

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Bayes Rule

$$\hat{c} = \ argmax_{c \in C} P(c|text)$$

$$\hat{c} = \ argmax_{c \in C} P(text|c) \times \frac{P(c)}{P(text)}$$

Naïve Bayes - Classification

$$\hat{c} = \ argmax_{c \in C} P(text|c) \times \frac{P(c)}{P(text)}$$

Naïve Bayes - Classification

$$\hat{c} = \ argmax_{c \in C} P(text|c) \times \frac{P(c)}{P(text)}$$

- The formula can be simplified by dropping de denominator $P(text)$.
- What does it mean?
  - We are actually calculating the probability of each class.
  - But for all classes $P(text)$ does not change, because it is always the same text.
    - We are always calculating the most likely class c for a same $text$.

Naïve Bayes - Classification

$$\hat{c} = \ argmax_{c \in C} P(text|c) \times \frac{P(c)}{P(text)}$$

- The formula can be simplified by dropping de denominator $P(text)$.
- What does it mean?
  - We are actually calculating the probability of each class.
  - But for all classes $P(text)$ does not change, because it is always the same text.
    - We are always calculating the most likely class c for a same $text$.
- Thus:

$$\hat{c} = \ argmax_{c \in C} P(text|c) P(c)$$

Naïve Bayes - Classification

- Calculate the most probable class $\hat{c}$ for the text.
- It is the one with the highest product between:
  - The **likelihood** of the text being of that given class $P(text|c)$
  - The **prior** probability of the class $P(c)$

$$\hat{c} = \ argmax_{c \in C} P(text|c)P(c)$$

Naïve Bayes - Classification

$$\hat{c} = \ argmax_{c \in C} P(text|c)P(c)$$

- We can represent the text as a set of words or tokens. $P(text|c)$ then becomes a probability of those words. Those words represent our *features*.

$$\hat{c} = \ argmax_{c \in C} P(w_1, w_2, \ldots, w_n|c)P(c)$$

Naïve Bayes - Classification

- We can represent the text as a set of words or tokens. $P(text|c)$ then becomes a probability of those words. Those words represent our *features*.

$$\hat{c} = \ argmax_{c \in C} P(w_1, w_2, \ldots, w_n|c)P(c)$$

- The Naïve Bayes assumption: the probabilities of each feature are independent given a class, therefore:

$$P(w_1, w_2, \ldots, w_n|c) = P(w_1|c)P(w_2|c) \ldots P(\ w_n|c)$$

Naïve Bayes - Classification

$$\hat{c} = \ argmax_{c \in C} P(w_1, w_2, \ldots, w_n | c) P(c)$$

$$P(w_1, w_2, \ldots, w_n | c) = P(w_1 | c) P(w_2 | c) \ldots P(w_n | c)$$

Thus:

$$\hat{c} = \ argmax_{c \in C} P(c) \prod_{w \in text} P(w_i | c)$$

Naïve Bayes – Sentiment Analysis

Negative:
"We are afraid of the economy recession."
"People are afraid of losing their jobs."

Positive:
"The economy is recovering."
"People are getting new jobs."

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we		
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}		

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	$\frac{1}{\quad}$	0
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}	14	9

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	$\frac{1}{14}$	0
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}	14	9

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	0.07	0
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}	14	9

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	0.07	0
are	0.14	0.11
afraid	0.14	0
of	0.14	0
the	0.07	0.11
economy	0.07	0.11
recession	0.07	0
people	0.07	0.11
losing	0.07	0
their	0.07	0
jobs	0.07	0.11
is	0	0.11
recovering	0	0.11
getting	0	0.11
new	0	0.11
N_{class}	1	1

P(word|class)

Vocabulary	Neg	Pos
we	0.07	0
are	0.14	0.11
afraid	0.14	0
of	0.14	0
the	0.07	0.11
economy	0.07	0.11
recession	0.07	0
people	0.07	0.11
losing	0.07	0
their	0.07	0
jobs	0.07	0.11
is	0	0.11
recovering	0	0.11
getting	0	0.11
new	0	0.11
N_{class}	1	1

We can get a hint that the data is far from great quality.

Why is that?

We'll discuss it further.

P(word|class)

Vocabulary	Neg	Pos
we	0.07	0
are	0.14	0.11
afraid	0.14	0
of	0.14	0
the	0.07	0.11
economy	0.07	0.11
recession	0.07	0
people	0.07	0.11
losing	0.07	0
their	0.07	0
jobs	0.07	0.11
is	0	0.11
recovering	0	0.11
getting	0	0.11
new	0	0.11
N_{class}	1	1

Neutral words have similar probabilities.

P(word|class)

Vocabulary	Neg	Pos
we	0.07	0
are	0.14	0.11
afraid	0.14	0
of	0.14	0
the	0.07	0.11
economy	0.07	0.11
recession	0.07	0
people	0.07	0.11
losing	0.07	0
their	0.07	0
jobs	0.07	0.11
is	0	0.11
recovering	0	0.11
getting	0	0.11
new	0	0.11
N_{class}	1	1

Zeros are problematic.
No way to compare.

Use smoothing.

Laplacian Smoothing

- Traditional probability formula:

$$P(word_i|class) = \frac{count(word_i, class)}{N_{class}}$$

- If the word is not present in the training for the class, the probability will be zero. We use smoothing to avoid that.

$$P(word_i|class) = \frac{count(word_i, class) + 1}{N_{class} + |V|}$$

- Where $|V|$ is the number of unique words in the vocabulary.

P(word|class) - Smoothing

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we		
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}		

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	$\frac{1+1}{}$	0
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}	14	9

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	$\frac{1+1}{14+14}$	0
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}	14	9

P(word|class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	0.07	0
are		
afraid		
of		
the		
economy		
recession		
people		
losing		
their		
jobs		
is		
recovering		
getting		
new		
N_{class}	14	9

P(word|class) - Smoothing

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	2	0
the	1	1
economy	1	1
recession	1	0
people	1	1
losing	1	0
their	1	0
jobs	1	1
is	0	1
recovering	0	1
getting	0	1
new	0	1
N_{class}	14	9

Vocabulary	Neg	Pos
we	0.071	0.043
are	0.104	0.086
afraid	0.104	0.043
of	0.104	0.043
the	0.071	0.086
economy	0.071	0.086
recession	0.071	0.043
people	0.071	0.086
losing	0.071	0.043
their	0.071	0.043
jobs	0.071	0.086
is	0.035	0.086
recovering	0.035	0.086
getting	0.035	0.086
new	0.035	0.086
N_{class}	1	1

Naïve Bayes

- Naïve Bayes inference condition rule for binary classification:

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

- If the result is >1, the classification is positive.
- If the result is <1, the classification is negative.

Naïve Bayes

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

- Example:

"The economy is recovering."

Naïve Bayes

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

- Example:

"The economy is recovering."

$$\frac{0.086}{0.071} *$$

Naïve Bayes

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

- Example:

"The economy is recovering."

$$\frac{0.086}{0.071} * \frac{0.086}{0.035} *$$

Naïve Bayes

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

- Example:

"The economy is recovering."

$$\frac{0.086}{0.071} * \frac{0.086}{0.035} * \frac{0.086}{0.035}$$

Naïve Bayes

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

- Example:

"The economy is recovering."

$$\frac{0.086}{0.071} * \frac{0.086}{0.035} * \frac{0.086}{0.035} * \frac{0.086}{0.035}$$

Naïve Bayes

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

- Example:

"The economy is recovering."

$$\frac{0.086}{0.071} * \frac{0.086}{0.071} * \frac{0.086}{0.035} * \frac{0.086}{0.035} \approx 8.85$$

Positive!

Naïve Bayes

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

- Example:

"People are afraid of losing their jobs."

Naïve Bayes

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

- Example:

"People are afraid of losing their jobs."

$$\frac{0.086}{0.071} * \frac{0.086}{0.104} * \frac{0.043}{0.104} * \frac{0.043}{0.104} * \frac{0.043}{0.071} * \frac{0.043}{0.071} * \frac{0.086}{0.071} \approx 0.076$$

Negative!

Likelihood - Ratio

- Words can have many levels of emotion intensity.
- For our example purpose, words can think of three levels:
  - Positive, Negative or Neutral.
- We can categorize them by their conditional probabilities.

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

- With this ratio value, we can represent the emotion of a word with a single value that ranges from Negative to Positive, passing by Neutral.

Likelihood - Ratio

$$ratio(w) = \frac{P(w|pos)}{P(w|neg)} \approx \frac{freq(w,1)+1}{freq(w,0)+1}$$

Vocabulary	Pos	Neg	ratio
we	0.043	0.071	0.60
are	0.086	0.104	0.82
afraid	0.043	0.104	0.41
of	0.043	0.104	0.41
the	0.086	0.071	1.21
economy	0.086	0.071	1.21
recession	0.043	0.071	0.60
people	0.086	0.071	1.21
losing	0.043	0.071	0.60
their	0.043	0.071	0.60
jobs	0.086	0.071	1.21
is	0.086	0.035	2.45
recovering	0.086	0.035	2.45
getting	0.086	0.035	2.45
new	0.086	0.035	2.45
Sum	1	1	-

Likelihood

- On Naïve Bayes those ratios are the basis of binary classification.

- Earlier we had Positive if > 1 and negative if < 1:

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

Likelihood

- On Naïve Bayes those rations are the basis of binary classification.

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

Likelihood

Likelihood

- On Naïve Bayes those rations are the basis of binary classification.
- If we take a ratio between the positive and negative texts, we have:

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

Prior ratio

Likelihood

- On Naïve Bayes those rations are the basis of binary classification.
- The prior ratio is necessary because we don't have the same amount of negative and positive texts in the corpus.
- Important with unbalanced datasets.

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

Prior ratio

Log Likelihood

- As we could see on our example tables, we use lots of numbers < 1 with potentially many decimal cases.
- Computers are not good on storing very small values (underflow with multiplications).
- We can loose a lot of accuracy because of that.
- How to handle that issue?

Log Likelihood

- How to handle that issue?
- A property of logarithms:

$$\log(a * b) = \log(a) + log(b)$$

- Remember our formula:

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

Log Likelihood

- The trick is to take the log:

$$log\left(\frac{P(pos)}{P(neg)}\prod_{i=1}^{m}\frac{P(word_i|pos)}{P(word_i|neg)}\right)$$

- And then write is as:

$$log\left(\frac{P(pos)}{P(neg)}\right) + \log\left(\prod_{i=1}^{m}\frac{P(word_i|pos)}{P(word_i|neg)}\right)$$

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

log prior + log likelihood

Calculation

- Example (not the same data as before, for example only)

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

Vocabulary	Pos	Neg	λ
we	0.05	0.05	
are	0.05	0.05	
recovering	0.09	0.01	
from	0.01	0.01	
the	0.01	0.01	
economic	0.01	0.01	
recession	0.02	0.09	
people	0.01	0.01	
unemployed	0.01	0.09	

Calculation

- Example (not the same data as before, for example only)

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

$$\lambda(we) = log \frac{0.05}{0.05} = \log(1) = 0$$

Vocabulary	Pos	Neg	λ
we	0.05	0.05	
are	0.05	0.05	
recovering	0.09	0.01	
from	0.01	0.01	
the	0.01	0.01	
economic	0.01	0.01	
recession	0.02	0.09	
people	0.01	0.01	
unemployed	0.01	0.09	

Calculation

- Example (not the same data as before, for example only)

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

$$\lambda(we) = log \frac{0.05}{0.05} = \log(1) = 0$$

In that case, 0 is neutral

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	
recovering	0.09	0.01	
from	0.01	0.01	
the	0.01	0.01	
economic	0.01	0.01	
recession	0.02	0.09	
people	0.01	0.01	
unemployed	0.01	0.09	

Calculation

- Example (not the same data as before, for example only)

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

$$\lambda(recovering) = log \frac{0.09}{0.01} = 2.2$$

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	
from	0.01	0.01	
the	0.01	0.01	
economic	0.01	0.01	
recession	0.02	0.09	
people	0.01	0.01	
unemployed	0.01	0.09	

Calculation

- Example (not the same data as before, for example only)

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

$$\lambda(recovering) = log \frac{0.09}{0.01} = 2.2$$

Positive (λ >0)

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	
the	0.01	0.01	
economic	0.01	0.01	
recession	0.02	0.09	
people	0.01	0.01	
unemployed	0.01	0.09	

Calculation

- Example (not the same data as before, for example only)

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

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Negative words have $\lambda < 0$

Calculation

- Example (not the same data as before, for example only)

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

REMEMBER: we use the natural log.

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

- 0

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

- 0 + 0

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

- 0 + 0 + 2.2

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

- 0 + 0 + 2.2 + 0

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

- 0 + 0 + 2.2 + 0 + 0 = 2.2 + logprior

$$\frac{P(pos)}{P(neg)}$$

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

- 0 + 0 + 2.2 + 0 + 0 = 2.2 + 0

Logprior = 0 because we have a balanced dataset.

Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Calculation

- Example (not the same data as before, for example only)
- "We are recovering from recession."

$$\sum_{i=1}^{m} \lambda(word_i)$$

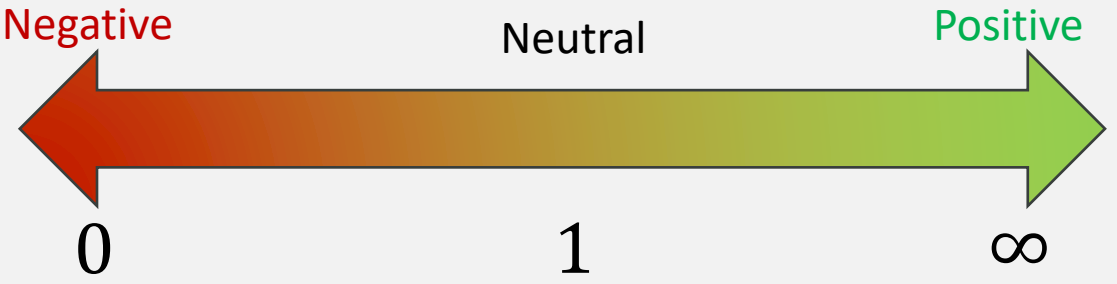
- 0 + 0 + 2.2 + 0 + 0 = 2.2

  Positive

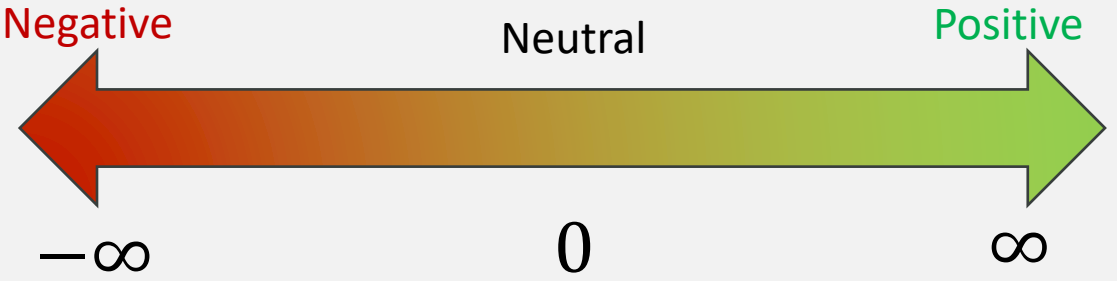
Vocabulary	Pos	Neg	λ
we	0.05	0.05	0
are	0.05	0.05	0
recovering	0.09	0.01	2.2
from	0.01	0.01	0
the	0.01	0.01	0
economic	0.01	0.01	0
recession	0.02	0.09	-1.5
people	0.01	0.01	0
unemployed	0.01	0.09	-2.2

Classification

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



$$\sum_{i=1}^{n} log\left(\frac{P(word_i|pos)}{P(word_i|neg)}\right) > 0$$



Naïve Bayes Training

- Differently from other Machine Learning models.
- No gradient descent.
- Naïve Bayes counts frequencies of the words in the corpus.

Naïve Bayes Training

Naïve Bayes Training

Negative:
"We are afraid of the economy recession."
"People are afraid of losing their jobs."

Positive:
"The economy is recovering."
"People are getting new jobs."

Naïve Bayes Training

- Lowercase
- Remove punctuation, urls, etc
- Remove stop words
- Stemming
- Tokenize sentences
- …

Naïve Bayes Training

- Unknown Words
  - Ignore them, no probability at all
  - Using <UNK> token does not help (becomes general word)
- Stopwords
  - Normally doesn’t help

Naïve Bayes Training

freq(w, class)

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	1	0
the	1	1
recession	2	1
…	…	…

Naïve Bayes Training

Vocabulary	Neg	Pos
we	1	0
are	2	1
afraid	2	0
of	1	0
the	1	1
recession	2	1
…	…	…

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

(Laplacian smoothing)

Vocabulary	Neg	Pos
we	0.071	0.043
are	0.104	0.086
afraid	0.104	0.043
of	0.071	0.043
the	0.071	0.086
recession	0.104	0.086
…	…	…

Naïve Bayes Training

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

Vocabulary	Neg	Pos	λ
we	0.071	0.043	-0.501
are	0.104	0.086	0.190
afraid	0.104	0.043	0.883
of	0.071	0.043	-0.501
the	0.071	0.086	-0.191
recession	0.104	0.086	0.190
…	…	…	…

Naïve Bayes Training

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

D_{pos}: # positive texts

D_{neg}: # negative texts

For a balanced dataset, logprior = 0

Testing Naïve Bayes

- Evaluate performance on unseen data (X_{val}, Y_{val}).
- Use λ and logprior for each new text.
- Compute Accuracy $(pred == Y_{val})$.
- Words not present in the vocabulary don't have a λ.
  - Considered neutral words (value λ=0).

Testing Naïve Bayes

$$score = (X_{val}, \lambda, logprior)$$
$$pred = score > 0$$

$$\begin{bmatrix} 0.4 \\ -1.2 \\ 0.9 \\ \dots \\ score_m \end{bmatrix} > 0 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \dots \\ pred_m \end{bmatrix}$$

$$accuracy = \sum_{i=1}^{m} \frac{pred_i == yval_i}{m}$$

Testing Naïve Bayes

$$score = (X_{val}, \lambda, logprior)$$
$$pred = score > 0$$

$$\begin{bmatrix} 0.4 \\ -1.2 \\ 0.9 \\ \dots \\ score_m \end{bmatrix} > 0 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \dots \\ pred_m \end{bmatrix}$$

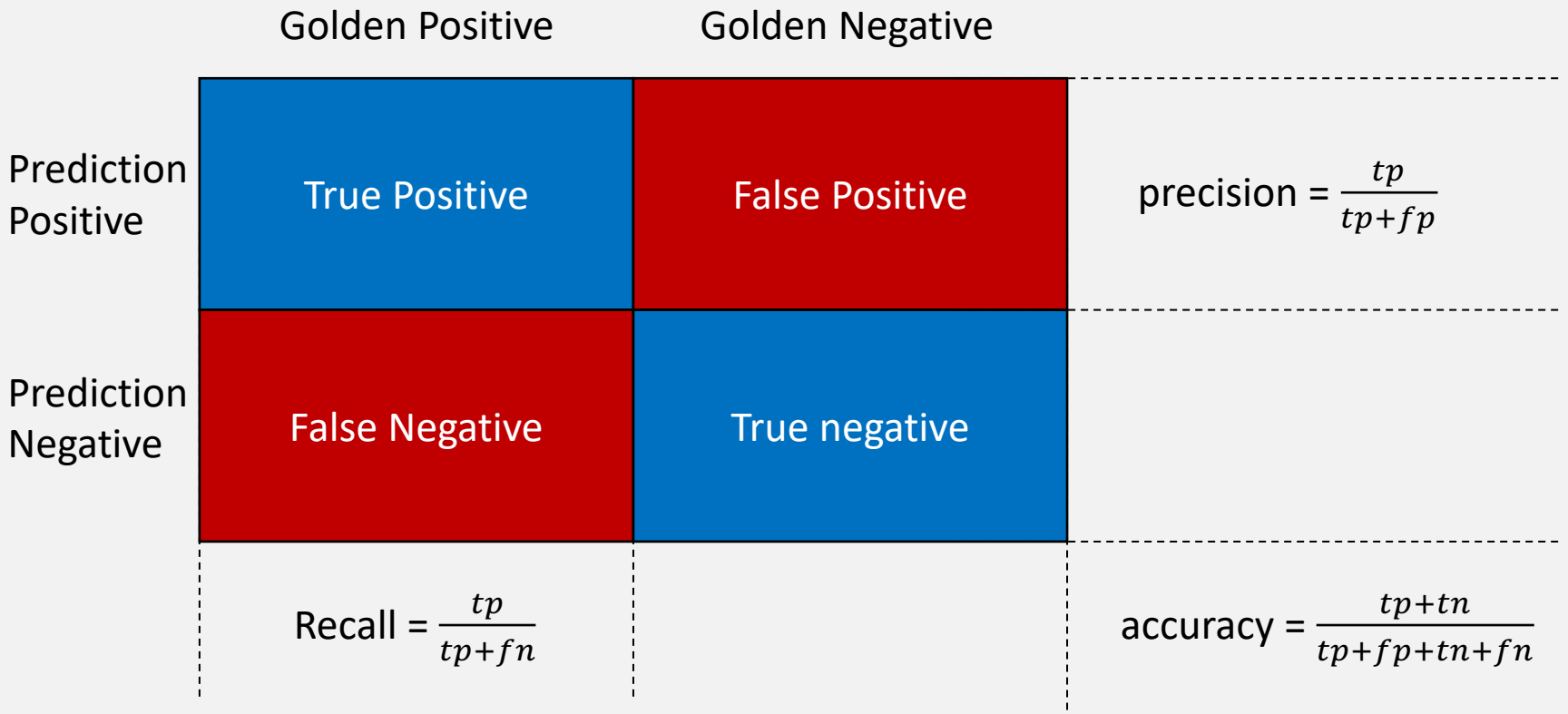
$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ Y_{val_m} \end{bmatrix} == \begin{bmatrix} 1 \\ 0 \\ 1 \\ \dots \\ pred_m \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ Y_{val_m} == pred_m \end{bmatrix}$$

$$accuracy = \sum_{i=1}^{m} \frac{pred_i == yval_i}{m}$$

Testing Naïve Bayes

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



Recall measures the percentage of items actually present in the input that were correctly identified by the system.

Testing Naïve Bayes

- F-Score: combines Precision and Recall.

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

- β favors Recall or Precision
  - β > 1 favors recall
  - β < 1 favors precision

- We mostly use β = 1, the F_1-score

Naïve Bayes Limitations

- Assumption 1: words in a sentence are independent.
  - That is normally wrong.
  - The words that come before usually serve as a hint of the next one's meaning.
  - We will see more intelligent models in the following classes.
- Assumption 2: relies on the distribution of the training dataset.
  - In the real word, we usually don't have balanced datasets.
  - We end up having to synthetically balance the data.

Naïve Bayes Limitations

- Because of the assumptions, Naïve Bayes is usually **not** capable of:
- Detecting sarcasm, irony, and other adversarial sentiments.
  - Because it relies on independent words, and not how they are used.
- Considering the order of the words.
  - E.g.: I am not sad.
  - Might be classified as negative, because contains negative words, even if they are used to express the absence of a negative sentiment.

Summary

- Naïve Bayes can be used for any type of classification, not only as positive or negative texts as the example given
- E.g.: detecting spam emails.
- It is also simple and fast to implement and train.
- It has limitations but work very well, nonetheless.

Naïve Bayes – Possible Improvements

- Dealing with naiveness.
- Relies too much on data.
  - The counting tables need too many examples, very well labeled.

Naïve Bayes – Possible Improvements

- Dealing with naiveness:
  - I am not sad.
  - I am sad.
  - I am happy about it.
  - I am not happy about it.

Naïve Bayes considers words to be independent from each other.

They are not.

Naïve Bayes – Possible Improvements

- Dealing with naiveness:
  - I am not sad.
  - I am sad.
  - I am happy about it.
  - I am not happy about it.

Naïve Bayes considers words to be independent from each other.

They are not.

Simple fix: *et pluribus unum*

Naïve Bayes – Possible Improvements

- Dealing with naiveness:
  - I am not sad.
    - I am not NOT_sad.
  - I am sad.
  - I am happy about it.
  - I am not happy about it.
    - I am not NOT_happy NOT_about NOT_it.

Naïve Bayes considers words to be independent from each other.

They are not.

Simple fix: *et pluribus unum*

Naïve Bayes – Possible Improvements

- Relies too much on data.
  - The counting tables need too many examples, very well labeled.

Naïve Bayes – Possible Improvements

- Relies too much on data.
  - The counting tables need too many examples, very well labeled.
- Solution: Lexicons.
  - Some words are inherently positive or negative.
    - A characteristic from the language.
  - People have done it before. A lot.
  - There is a plethora of publicly available words lists (Lexicons) that represent sentiments.

Naïve Bayes – Possible Improvements

- Solution: Lexicons.
  - There is a plethora of publicly available words lists (Lexicons) that represent sentiments.
  - General Inquirer (Stone et al., 1966), LIWC (Pennebaker et al., 2007), the opinion lexicon General Inquirer LIWC of Hu and Liu (2004) and the MPQA Subjectivity Lexicon (Wilson et al., 2005).
- How to use it:
  - Create new features: “word is positive” and “word is negative”.
  - Using 2 features is not that good…
    - Keep it to extreme cases of very bad or lacking data.
    - Add to the known features.

Thank you!