Lecture 3:
n-gram LMs and Smoothing
+ Logistic Regression

Instructor: Swabha Swayamdipta

USC CSCI 544 Applied NLP

Sep 3, Fall 2024

Some slides adapted from Dan Jurafsky and Chris Manning and Xuezhe Ma

Lecture Outline

- Announcements + Recap
  - n-gram Language Models
  - Zeros!
- Smoothing
- Basics of Supervised Machine Learning
  - I. Data: Preprocessing and Feature Extraction
  - II. Model:
    - I. Logistic Regression
  - III. Loss
  - IV. Optimization Algorithm
  - V. Inference

Announcements
+
Recap

Logistics and Announcements

Logistics and Announcements

- Today: HW1 released
    - TA Office Hours: HW related questions

Logistics and Announcements

- Today: HW1 released
  - TA Office Hours: HW related questions
- My Office Hours: Questions about lectures, conceptual questions, class project ideas etc.

Logistics and Announcements

- Today: HW1 released
  - TA Office Hours: HW related questions
- My Office Hours: Questions about lectures, conceptual questions, class project ideas etc.
- Email policy: Please use Brightspace for your questions, which could be answered by TAs and might be beneficial for the rest of the class as well
  - Email me very sparingly, I will NOT be able to respond to everyone
    - I will especially NOT respond to questions regarding logistics, which are either discussed in class or mentioned on the website / Brightspace

Logistics and Announcements

- Today: HW1 released
  - TA Office Hours: HW related questions
- My Office Hours: Questions about lectures, conceptual questions, class project ideas etc.
- Email policy: Please use Brightspace for your questions, which could be answered by TAs and might be beneficial for the rest of the class as well
  - Email me very sparingly, I will NOT be able to respond to everyone
    - I will especially NOT respond to questions regarding logistics, which are either discussed in class or mentioned on the website / Brightspace
- CARC access for Project
  - Not possible to get for everyone
  - Please let us know if needed and depending on your project description we will apply for it
  - Not to be used for anything beyond your class project

Logistics and Announcements

- Today: HW1 released
  - TA Office Hours: HW related questions
- My Office Hours: Questions about lectures, conceptual questions, class project ideas etc.
- Email policy: Please use Brightspace for your questions, which could be answered by TAs and might be beneficial for the rest of the class as well
  - Email me very sparingly, I will NOT be able to respond to everyone
    - I will especially NOT respond to questions regarding logistics, which are either discussed in class or mentioned on the website / Brightspace
- CARC access for Project
  - Not possible to get for everyone
  - Please let us know if needed and depending on your project description we will apply for it
  - Not to be used for anything beyond your class project
- Thu: Quiz 1 (Bring your laptop!)

Logistics and Announcements

- Today: HW1 released
    - TA Office Hours: HW related questions
- My Office Hours: Questions about lectures, conceptual questions, class project ideas etc.
- Email policy: Please use Brightspace for your questions, which could be answered by TAs and might be beneficial for the rest of the class as well
    - Email me very sparingly, I will NOT be able to respond to everyone
        - I will especially NOT respond to questions regarding logistics, which are either discussed in class or mentioned on the website / Brightspace
- CARC access for Project
    - Not possible to get for everyone
    - Please let us know if needed and depending on your project description we will apply for it
    - Not to be used for anything beyond your class project
- Thu: Quiz 1 (Bring your laptop!)
- Final Exam on 12/5 (in person)

Logistics and Announcements

- Today: HW1 released
  - TA Office Hours: HW related questions
- My Office Hours: Questions about lectures, conceptual questions, class project ideas etc.
- Email policy: Please use Brightspace for your questions, which could be answered by TAs and might be beneficial for the rest of the class as well
  - Email me very sparingly, I will NOT be able to respond to everyone
    - I will especially NOT respond to questions regarding logistics, which are either discussed in class or mentioned on the website / Brightspace
- CARC access for Project
  - Not possible to get for everyone
  - Please let us know if needed and depending on your project description we will apply for it
  - Not to be used for anything beyond your class project
- Thu: Quiz 1 (Bring your laptop!)
- Final Exam on 12/5 (in person)
- Final Report Submission for Team Projects on 12/17 (online)

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, \ldots w_n)$$

A model that assigns probabilities to sequences of words is called a language model

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, \ldots w_n)$$

Related task: probability of an upcoming word: $P(w_n \mid w_1, w_2, w_3, w_4, \ldots w_{n-1})$

A model that assigns probabilities to sequences of words is called a language model

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

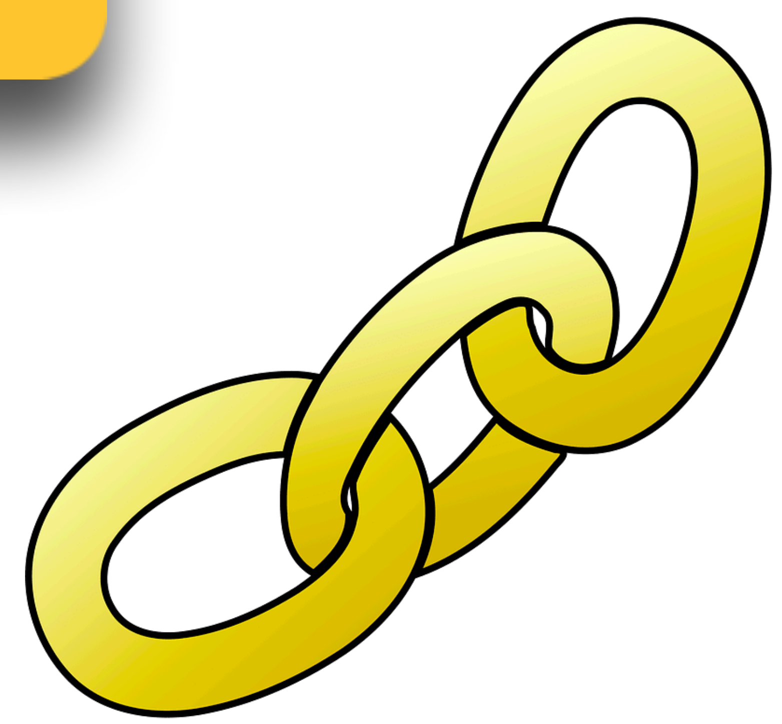
$$P(\mathbf{w}) = P(w_1, w_2, w_3, \dots w_n)$$

Related task: probability of an upcoming word: $P(w_n \mid w_1, w_2, w_3, w_4, \dots w_{n-1})$

A model that assigns probabilities to sequences of words is called a language model

Chain Rule

$$P(w_1, w_2, \dots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1} \dots w_1)$$



How to estimate the probability of the next word?

$$P(\text{that} \mid \text{its water is so transparent}) = \frac{Count(\text{its water is so transparent that})}{Count(\text{its water is so transparent})}$$

Maximum Likelihood Estimate

How to estimate the probability of the next word?

$$P(\text{that} \mid \text{its water is so transparent}) = \frac{Count(\text{its water is so transparent that})}{Count(\text{its water is so transparent})}$$

Maximum Likelihood Estimate

Too many possibilities to count! Too few sentences that look like this…

Need to make some simplifying assumptions…

Markov Assumption

$$P(w_i \mid w_1, w_2, \ldots w_{i-1}) \approx P(w_i \mid w_{i-k+1} \ldots w_{i-1})$$

k-th order Markov Assumption

In other words, we approximate each component in the product such that it is only conditioned on the previous $k-1$ elements

Markov Assumption

$$P(w_i \mid w_1, w_2, \ldots w_{i-1}) \approx P(w_i \mid w_{i-k+1} \ldots w_{i-1})$$

k-th order Markov Assumption

In other words, we approximate each component in the product such that it is only conditioned on the previous $k - 1$ elements

$$P(w_1, w_2, \ldots w_n) = \prod_i P(w_i \mid w_{i-k+1} \ldots w_{i-1})$$

n-gram models

n-gram models

Unigram Model

$$P(w_1, w_2, \ldots w_n) \approx \prod_i P(w_i)$$

n-gram models

Unigram Model

$$P(w_1, w_2, \ldots w_n) \approx \prod_i P(w_i)$$

Bigram Model

$$P(w_1, w_2, \ldots w_n) \approx \prod_i P(w_i \mid w_{i-1})$$

n-gram models

Unigram Model

$$P(w_1, w_2, \ldots w_n) \approx \prod_i P(w_i)$$

Bigram Model

$$P(w_1, w_2, \ldots w_n) \approx \prod_i P(w_i \mid w_{i-1})$$

k-gram Model

$$P(w_1, w_2, \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k+1} \ldots w_{i-1})$$


You think you just fell
out of a coconut tree?


YOU THINK YOU JUST FELL OUT
OF A
COCONUT
TREE?
YOU EXIST IN THE CONTEXT OF ALL IN WHICH
YOU LIVE AND WHAT CAME BEFORE YOU.

Definitely true for tokens in natural language!

n-gram Models: Limitations

n-gram Models: Limitations

In general this is an insufficient model of language

- “The computer which I had just put into the machine room on the fifth floor crashed.”

n-gram Models: Limitations

In general this is an insufficient model of language

- "The computer which I had just put into the machine room on the fifth floor crashed."

Language has long-distance dependencies

n-gram Models: Limitations

In general this is an insufficient model of language

- "The computer which I had just put into the machine room on the fifth floor crashed."

At times the dependencies are not even clear!

- "The complex houses married and single soldiers and their families."

Language has long-distance dependencies

n-gram Models: Limitations

In general this is an insufficient model of language

- "The computer which I had just put into the machine room on the fifth floor crashed."

At times the dependencies are not even clear!

- "The complex houses married and single soldiers and their families."
- "The horse raced past the barn fell."

Language has long-distance dependencies

n-gram Models: Limitations

In general this is an insufficient model of language

- “The computer which I had just put into the machine room on the fifth floor crashed.”

At times the dependencies are not even clear!

- “The complex houses married and single soldiers and their families.”
- “The horse raced past the barn fell.”
- “The old man the boat.”

Language has long-distance dependencies

n-gram Models: Limitations

In general this is an insufficient model of language

- "The computer which I had just put into the machine room on the fifth floor crashed."

At times the dependencies are not even clear!

- "The complex houses married and single soldiers and their families."
- "The horse raced past the barn fell."
- "The old man the boat."

Garden Path Sentences

Language has long-distance dependencies

n-gram Models: Limitations

In general this is an insufficient model of language

- “The computer which I had just put into the machine room on the fifth floor crashed.”

At times the dependencies are not even clear!

- “The complex houses married and single soldiers and their families.”
- “The horse raced past the barn fell.”
- “The old man the boat.”

Garden Path Sentences

But we can often get away with n-gram models

Language has long-distance dependencies

Estimating bigram probabilities

Maximum Likelihood Estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Estimating bigram probabilities

Maximum Likelihood Estimate

Counts are whole numbers

$$P_{MLE}(w_i | w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Estimating bigram probabilities

Maximum Likelihood Estimate

Counts are whole numbers

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1} w_i)}{c(w_{i-1})}$$

Special edge case tokens: <s> and </s> for beginning of sentence and end of sentence, respectively

Estimating bigram probabilities

Maximum Likelihood Estimate

Counts are whole numbers

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

We do everything in log space to handle overflow issues

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Special edge case tokens: <s> and </s> for beginning of sentence and end of sentence, respectively

For the 9222 sentences in the Berkeley Restaurant Corpus:

For the 9222 sentences in the Berkeley Restaurant Corpus:

Unigram Counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

For the 9222 sentences in the Berkeley Restaurant Corpus:

Unigram Counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigram Counts

Next Word

History

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

For the 9222 sentences in the Berkeley Restaurant Corpus:

Unigram Counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigram Counts

Next Word

History

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Bigram Probabilities

w_i

w_{i-1}

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

For the 9222 sentences in the Berkeley Restaurant Corpus:

Unigram Counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigram Counts

Next Word

History

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Most n-grams are never seen!

Bigram Probabilities

w_i

w_{i-1}

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

How good is a language model?

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs
- returns the highest probability when evaluated on an unseen test set

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs
- returns the highest probability when evaluated on an unseen test set
  - Mantra: *I will never train my model on a test set*

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs
- returns the highest probability when evaluated on an unseen test set
    - Mantra: *I will never train my model on a test set*

Perplexity

$$PPL(\mathbf{w}) = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}}$$

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs
- returns the highest probability when evaluated on an unseen test set
  - Mantra: I will never train my model on a test set

Perplexity

$$PPL(\mathbf{w}) = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}}$$
$$= \exp(-\frac{1}{N} \log P(w_1, w_2, \ldots, w_N))$$

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs
- returns the highest probability when evaluated on an unseen test set
    - Mantra: *I will never train my model on a test set*

Perplexity

$$PPL(\mathbf{w}) = P(w_1, w_2, \ldots, w_N)^{\boxed{-\frac{1}{N}}}$$

Normalization Factor

$$= \exp(-\frac{1}{N} \log P(w_1, w_2, \ldots, w_N))$$

How good is a language model?

A better model of a text

- is one which assigns a higher probability to the word that actually occurs
- returns the highest probability when evaluated on an unseen test set
  - Mantra: *I will never train my model on a test set*

Perplexity

$$PPL(\mathbf{w}) = P(w_1, w_2, \ldots, w_N)^{\boxed{-\frac{1}{N}}}$$

Normalization Factor

$$= \exp\left(\boxed{-\frac{1}{N} \log P(w_1, w_2, \ldots, w_N)}\right)$$

Negative log likelihood

USC Viterbi

Bigram Perplexity

$$PPL(\mathbf{w}) = \exp(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid w_{i-1}))$$

Bigram Perplexity

$$PPL(\mathbf{w}) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i \mid w_{i-1})\right)$$

Lower the perplexity, better the language model

Bigram Perplexity

$$PPL(\mathbf{w}) = \exp(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i \mid w_{i-1}))$$

Lower the perplexity, better the language model

WSJ Perplexities

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Bigram Perplexity

$$PPL(\mathbf{w}) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i \mid w_{i-1})\right)$$

Lower the perplexity, better the language model

WSJ Perplexities

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

n-grams do a better and better job of modeling the training corpus as we increase the value of *n*

How best to evaluate an LM?

How best to evaluate an LM?

- Extrinsic evaluation
  - On an external task (e.g. summarization) that uses an LM
  - More reliable
  - Can be time-consuming; hard to design
    - Which is the best task? How many tasks to try?

How best to evaluate an LM?

- Extrinsic evaluation
  - On an external task (e.g. summarization) that uses an LM
  - More reliable
  - Can be time-consuming; hard to design
    - Which is the best task? How many tasks to try?
- Therefore, we often use intrinsic evaluation: perplexity
  - Bad approximation (less reliable)
    - Unless the test data looks just like the training data
  - Generally only useful in pilot experiments (faster to compute)

Generating from a bigram model

Generating from a bigram model

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
```

```
I want to eat Chinese food
```

USC Viterbi

Generating from a bigram model

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food </s>
```

I want to eat Chinese food

On your own: Sampling from a probability distribution

Shakespearean n-grams

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.

The WSJ is no Shakespeare!

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

The WSJ is no Shakespeare!

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Shakespearean corpus cannot produce WSJ vocabulary and vice versa

The WSJ is no Shakespeare!

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Shakespearean corpus cannot produce WSJ vocabulary and vice versa

Overfitting!

Two Types of Overfitting Issues

Two Types of Overfitting Issues

- At test time:

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts
  - May lead to undefined n-gram probabilities and perplexity

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts
  - May lead to undefined n-gram probabilities and perplexity
  - To be expected, very common!

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts
  - May lead to undefined n-gram probabilities and perplexity
  - To be expected, very common!
- Solutions:

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts
  - May lead to undefined n-gram probabilities and perplexity
  - To be expected, very common!
- Solutions:
  - Zero unigram counts: <UNK> token

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts
  - May lead to undefined n-gram probabilities and perplexity
  - To be expected, very common!
- Solutions:
  - Zero unigram counts: <UNK> token
    - Closed and Open Vocabularies

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero bi-gram counts
  - May lead to undefined n-gram probabilities and perplexity
  - To be expected, very common!
- Solutions:
  - Zero unigram counts: <UNK> token
    - Closed and Open Vocabularies
  - Zero bi-gram counts: Smoothing

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

N-gram models: Zero Counts

N-gram models: Zero Counts

- At test time, we may encounter tokens never seen (unigram with 0 frequency)
  - Very severe yet common problem resulting in undefined probabilities
  - Happens because of new terms, words, different dialects, evolving language
  - These are known as **OOV** for “out of vocabulary”, or <UNK> for **unknown tokens**

N-gram models: Zero Counts

- At test time, we may encounter tokens never seen (unigram with 0 frequency)
  - Very severe yet common problem resulting in undefined probabilities
  - Happens because of new terms, words, different dialects, evolving language
  - These are known as **OOV** for "out of vocabulary", or <UNK> for **unknown tokens**
- **Solution:** During training (probability estimation), replace all words that occur fewer than n times in the training set, where n is some small number by <UNK> and re-estimate the counts and probabilities.
  - At test time, any OOV token is automatically mapped to <UNK>

N-gram models: Zero Counts

- At test time, we may encounter tokens never seen (unigram with 0 frequency)
  - Very severe yet common problem resulting in undefined probabilities
  - Happens because of new terms, words, different dialects, evolving language
  - These are known as **OOV** for "out of vocabulary", or <UNK> for **unknown tokens**
- **Solution:** During training (probability estimation), replace all words that occur fewer than n times in the training set, where n is some small number by <UNK> and re-estimate the counts and probabilities.
  - At test time, any OOV token is automatically mapped to <UNK>
- Design: Open Vocabulary vs. Closed Vocabulary
  - Closed Vocabulary: predetermine the vocabulary (e.g. using a dictionary)
    - Restricted…why?
  - Open Vocabulary: no predetermination but anticipate new tokens

Open vs. Closed Vocabularies

Smoothing

Intuition for Smoothing

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

→

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

Intuition for Smoothing

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

→

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too

Intuition for Smoothing

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.	P(next word = *pizza* | previous word = *eat*) = 2/3 P(next word = *cake* | previous word = *eat*) = 1/3 All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $|V_{\text{bigrams}}| = ?$

Intuition for Smoothing

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

→

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $\quad\quad |V_{\text{bigrams}}| = ?$
- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words

Intuition for Smoothing

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

→

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $|V_{\text{bigrams}}| = ?$
- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words
- We want to **smooth the distribution from our counts**

Intuition for Smoothing

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too. →

P(next word = *pizza* | previous word = *eat*) = 2/3
P(next word = *cake* | previous word = *eat*) = 1/3
All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $|V_{\text{bigrams}}| = ?$
- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words
- We want to **smooth the distribution from our counts**

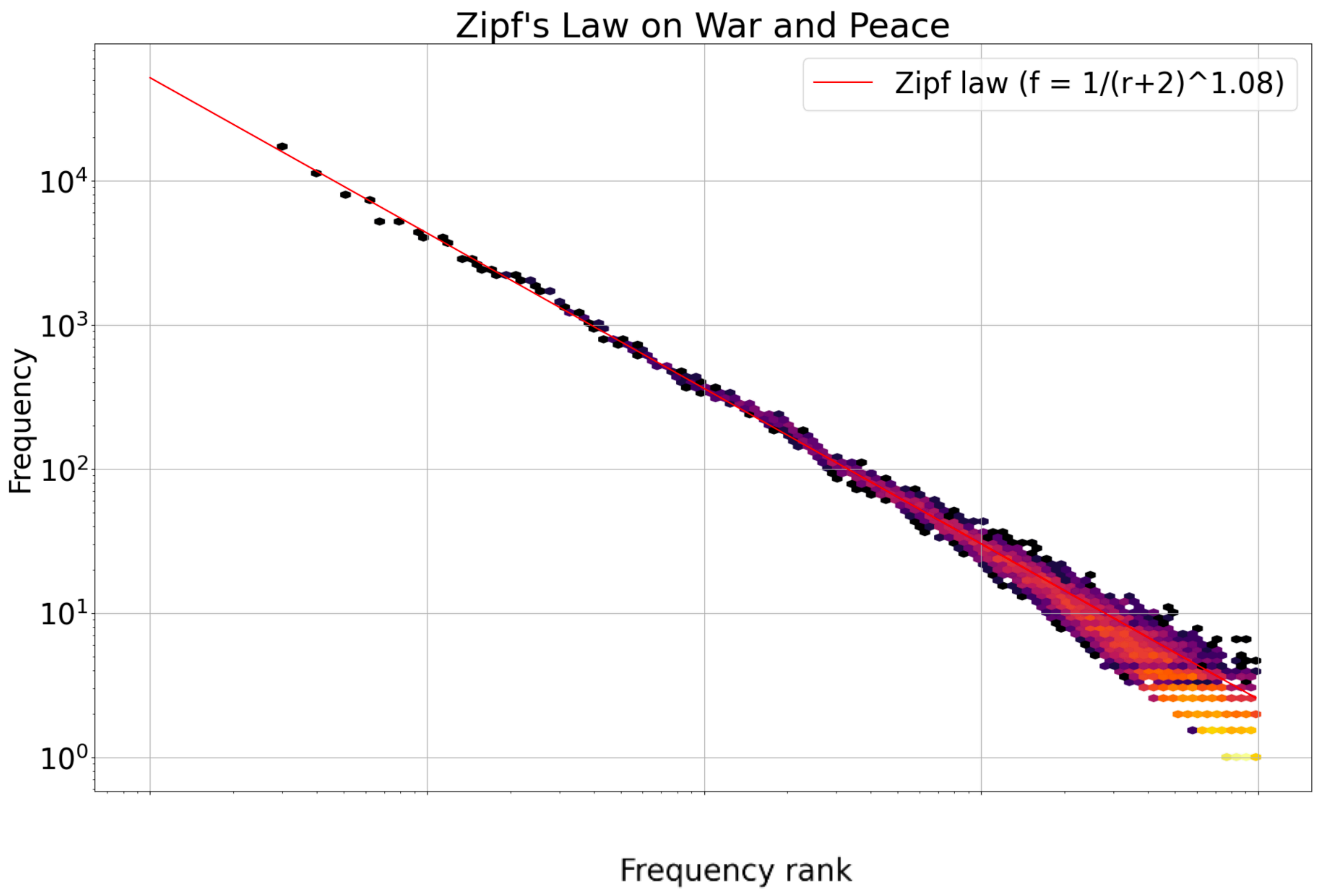
What does a count distribution look like?

Zipf's Law

The distribution over words resembles that of a power law:

- there will be a few words that are very frequent, and a long tail of words that are rare
- $freq_w(r) \approx r^{-s}$, where s is a constant

NLP algorithms must be especially robust to observations that do not occur or rarely occur in the training data

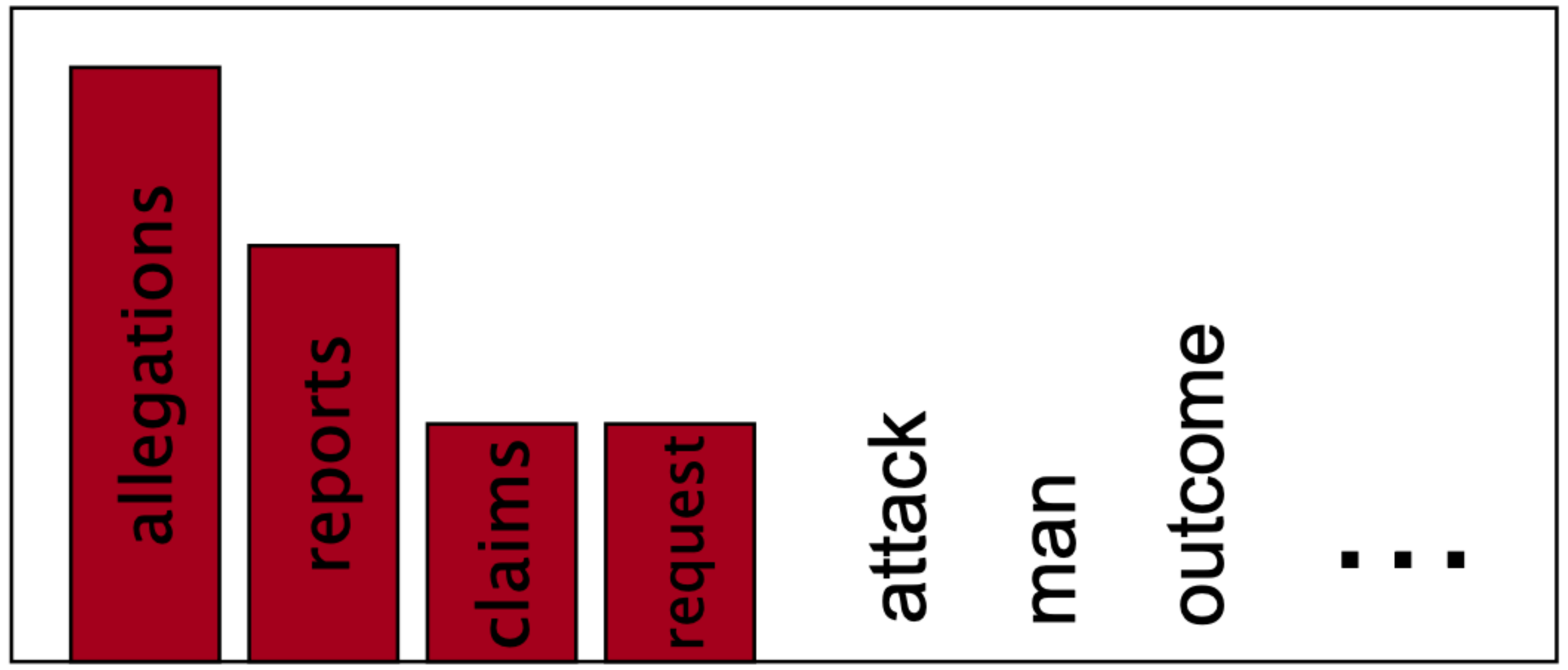


Zipf, G. K. (1949). Human behavior and the principle of least effort.

Smoothing ~ Massaging Probability Masses

When we have sparse statistics: $Count(w \mid \text{denied the})$

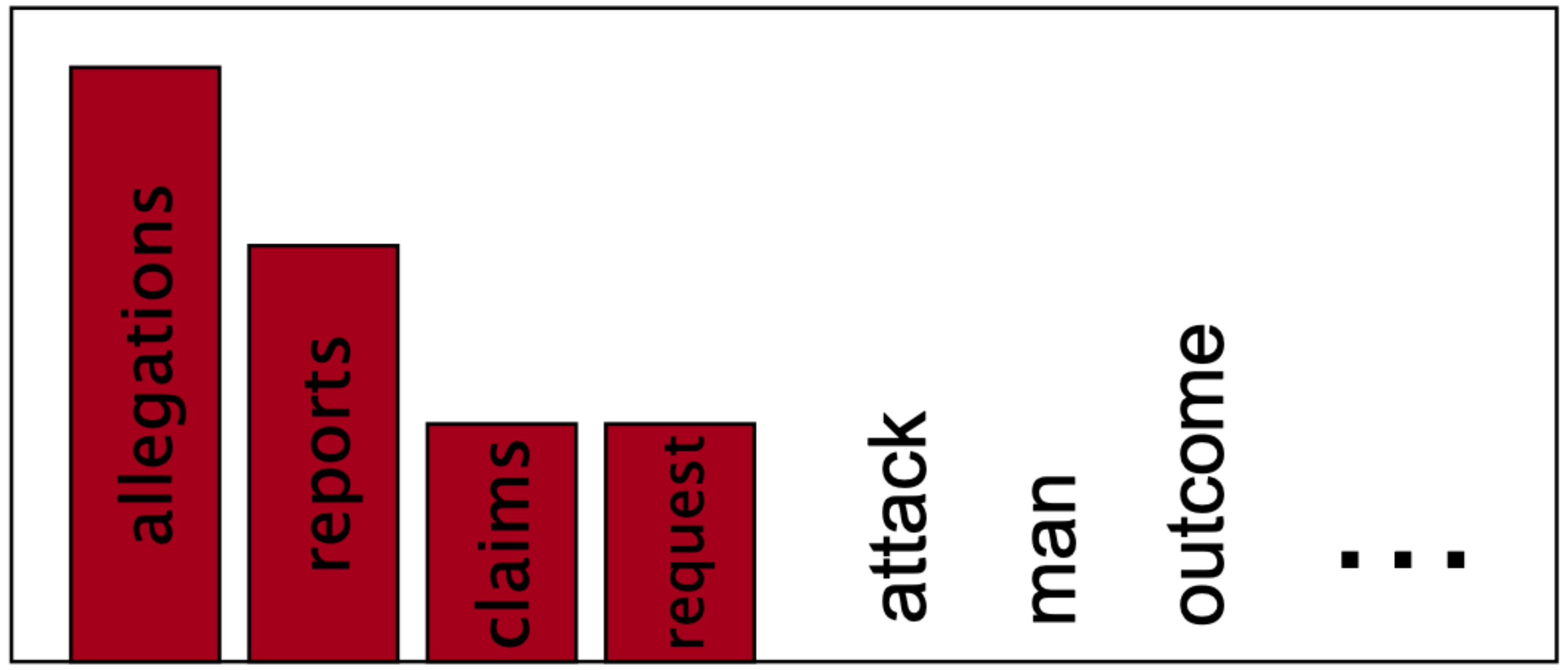
3 allegations
2 reports
1 claims
1 request
7 total



Smoothing ~ Massaging Probability Masses

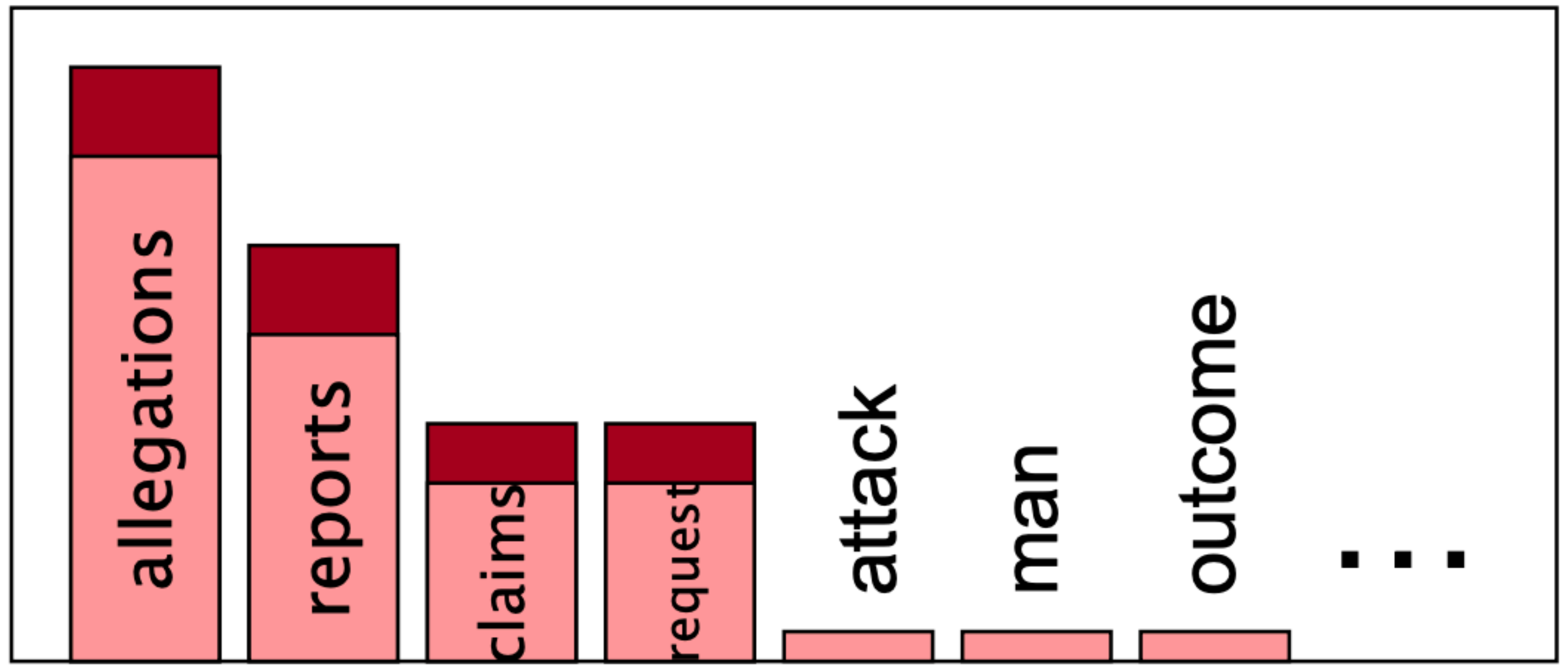
When we have sparse statistics: $Count(w \mid \text{denied the})$

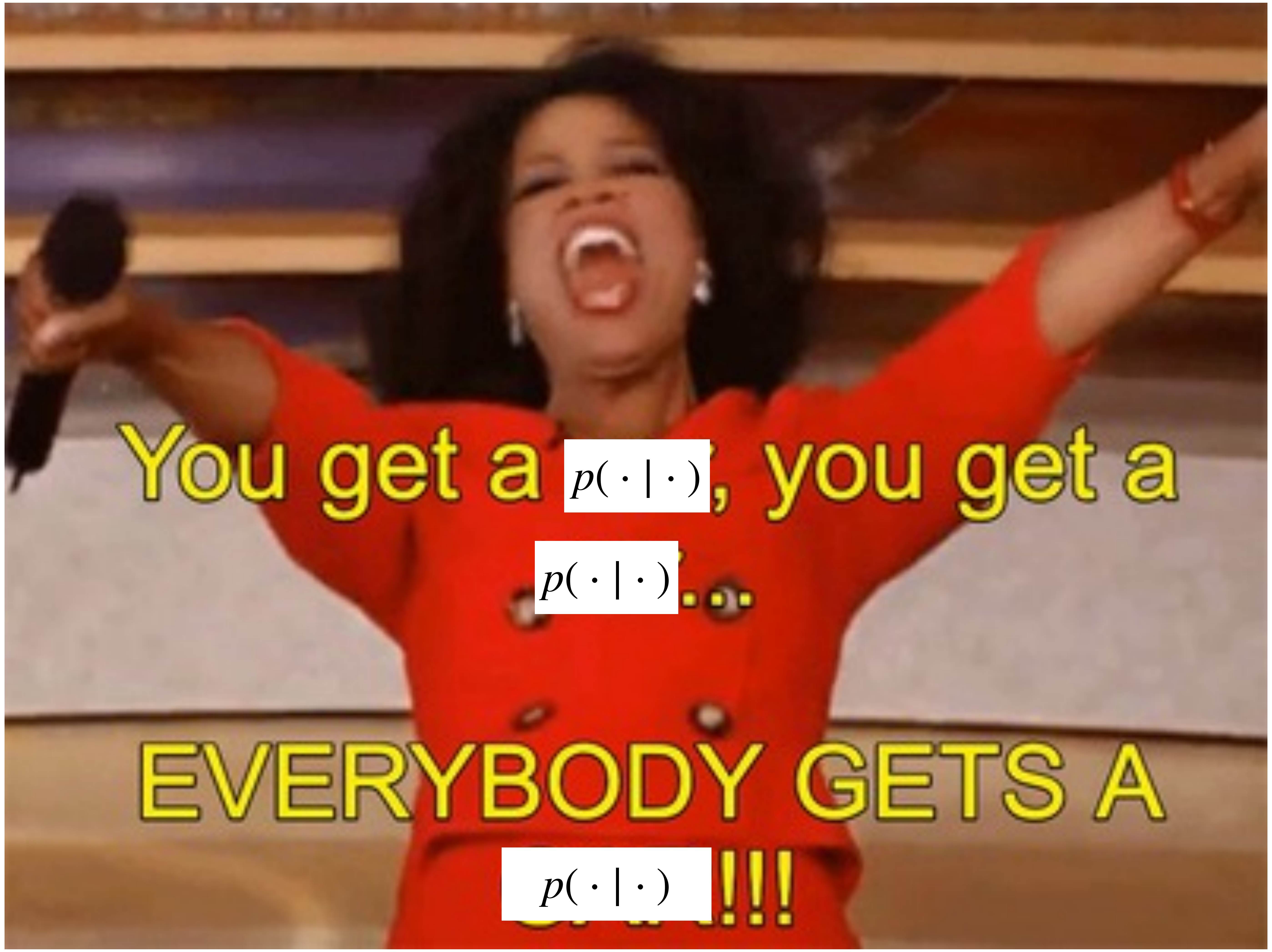
3 allegations
2 reports
1 claims
1 request
7 total



Steal probability mass to generalize better: $Count(w \mid \text{denied the})$

2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total




You get a $p(\cdot \mid \cdot)$, you get a
$p(\cdot \mid \cdot)$...
EVERYBODY GETS A
$p(\cdot \mid \cdot)$!!!

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

1. Pretend we saw each word one more time than we did

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!
3. All the counts that used to be zero will now have a count of 1…

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

Laplace smoothing

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!
3. All the counts that used to be zero will now have a count of 1…

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

Laplace smoothing

75 year old method!

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!
3. All the counts that used to be zero will now have a count of 1…

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

Laplace smoothing

75 year old method!

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!
3. All the counts that used to be zero will now have a count of 1…

Add-1 estimate

$$P_{Add-1}(w_i) = \frac{c(w_i)+1}{\sum_w (c(w)+1)} = \frac{c(w_i)+1}{V + \sum_w c(w)}$$

Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

Laplace smoothing

75 year old method!

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!
3. All the counts that used to be zero will now have a count of 1…

Add-1 estimate

$$P_{Add-1}(w_i) = \frac{c(w_i)+1}{\sum_w (c(w)+1)} = \frac{c(w_i)+1}{V+\sum_w c(w)}$$

What happens to our P if we don't increase the denominator?

Add-1 Estimation Bigrams

MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Pretend we saw each **bigram** one more time than we did

Add-1 Estimation Bigrams

MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Pretend we saw each **bigram** one more time than we did

Add-1 estimate

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) + 1}{c(w_{i-1}) + V}$$

Add-1 Estimation Bigrams

MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Pretend we saw each **bigram** one more time than we did

Add-1 estimate

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) + 1}{c(w_{i-1}) + V}$$

What does this do to the unigram counts?

Add-1 Estimation Bigrams

MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Pretend we saw each **bigram** one more time than we did

Add-1 estimate

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) + 1}{c(w_{i-1}) + V}$$

What does this do to the unigram counts?

Keep the same denominator as before and reconstruct bigram counts

Add-1 Estimation Bigrams

MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Pretend we saw each **bigram** one more time than we did

Add-1 estimate

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) + 1}{c(w_{i-1}) + V}$$

What does this do to the unigram counts?

Keep the same denominator as before and reconstruct bigram counts

$$= \frac{c^*(w_{i-1}w_i)}{c(w_{i-1})}$$

Recall: BRP Corpus

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Unigrams

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigrams

w_i

w_{i-1}

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Laplace-smoothed bigram counts

Just add one to all the counts!

USC Viterbi

Laplace-smoothed bigram counts

Just add one to all the counts!

w_i (columns); w_{i-1} (rows)

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigram probabilities

$$P_{Add-1}\left(w_i \mid w_{i-1}\right) = \frac{c(w_{i-1}w_i) + 1}{c(w_{i-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted Counts

$$c*(w_{i-1}w_i) = \frac{[c(w_{i-1}w_i)+1]c(w_{i-1})}{c(w_{i-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Compare with raw bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Compare with raw bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Big change to the counts!

Compare with raw bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Big change to the counts!

Perhaps 1 is too much, add a fraction?

Compare with raw bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Big change to the counts!

Perhaps 1 is too much, add a fraction?

Add-k smoothing

USC Viterbi

Compare with raw bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Big change to the counts!

Perhaps 1 is too much, add a fraction?

Add-k smoothing

k is a hyperparameter

Add-1 Estimation: Last thoughts

Add-1 Estimation: Last thoughts

So add-1 isn't used for n-grams, being something of a blunt instrument

Add-1 Estimation: Last thoughts

So add-1 isn't used for n-grams, being something of a blunt instrument 

Add-1 Estimation: Last thoughts

So add-1 isn't used for n-grams, being something of a blunt instrument 

- One-size-fits-all

Add-1 Estimation: Last thoughts

So add-1 isn't used for n-grams, being something of a blunt instrument

- One-size-fits-all

Add-1 is used to smooth other NLP models though…

- For text classification (Naïve Bayes)
- In domains where the number of zeros isn't so huge

Perhaps use some pre-existing evidence

Perhaps use some pre-existing evidence

- Condition on less context for contexts you haven’t learned much about

Interpolation

Perhaps use some pre-existing evidence

- Condition on less context for contexts you haven't learned much about

Interpolation

- mix unigram, bigram, trigram probabilities for a trigram LM

Interpolation

Perhaps use some pre-existing evidence

- Condition on less context for contexts you haven't learned much about

Interpolation

- mix unigram, bigram, trigram probabilities for a trigram LM
- mix n-gram, (n-1)-gram, … unigram probabilities for an n-gram LM

Interpolation

Perhaps use some pre-existing evidence

- Condition on less context for contexts you haven’t learned much about

Interpolation

- mix unigram, bigram, trigram probabilities for a trigram LM
- mix n-gram, (n-1)-gram, … unigram probabilities for an n-gram LM

Interpolation works better than Add-1 / Laplace

Linear Interpolation

Linear Interpolation

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1 P(w_i)$$
$$+\lambda_2 P(w_i \mid w_{i-1})$$
$$+\lambda_3 P(w_i \mid w_{i-2}w_{i-1})$$

Linear Interpolation

Simple Interpolation

$$\hat{P}(w_i \mid w_{i-2} w_{i-1}) = \lambda_1 P(w_i)$$
$$+\lambda_2 P(w_i \mid w_{i-1})$$
$$+\lambda_3 P(w_i \mid w_{i-2} w_{i-1})$$

USC Viterbi

Linear Interpolation

Simple Interpolation

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i \mid w_{i-2}w_{i-1})$$

$$\sum_k \lambda_k = 1$$

Linear Interpolation

Simple Interpolation

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i \mid w_{i-2}w_{i-1})$$

$$\sum_k \lambda_k = 1$$

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_3(w_{i-2}^{i-1})P(w_i \mid w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}^{i-1})P(w_i \mid w_{i-1}) + \lambda_1(w_{i-2}^{i-1})P(w_i)$$

USC Viterbi

Linear Interpolation

Simple Interpolation

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i \mid w_{i-2}w_{i-1})$$

$$\sum_k \lambda_k = 1$$

Context-Conditional Interpolation

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_3(w_{i-2}^{i-1}) P(w_i \mid w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}^{i-1}) P(w_i \mid w_{i-1}) + \lambda_1(w_{i-2}^{i-1}) P(w_i)$$

Different for every unique context

Linear Interpolation

Simple Interpolation

$$\hat{P}(w_i | w_{i-2}w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i | w_{i-2}w_{i-1})$$

$$\sum_k \boxed{\lambda_k} = 1$$

Hyperparameters!

Context-Conditional Interpolation

$$\hat{P}(w_i | w_{i-2}w_{i-1}) = \lambda_3(w_{i-2}^{i-1}) P(w_i | w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}^{i-1}) P(w_i | w_{i-1}) + \lambda_1(w_{i-2}^{i-1}) P(w_i)$$

Different for different bigrams!
Serve as Reconstituted Counts

Different for every unique context

How to set the λs?

How to set the λs?

Choose λs to maximize the probability of held-out data:

- Fix the n-gram probabilities (on the training data)
- Then search for λs that give largest probability to held-out set:

How to set the λs?

Choose λs to maximize the probability of held-out data:

- Fix the n-gram probabilities (on the training data)
- Then search for λs that give largest probability to held-out set:

$$logP(w_1 \ldots w_n | M(\lambda_1 \ldots \lambda_k)) = \sum_i logP_{M(\lambda_1 \ldots \lambda_k)}(w_i | w_{i-1})$$

n-grams Today

Infini-gram: Scaling Unbounded *n*-gram Language Models to a Trillion Tokens

Jiacheng Liu[♡] **Sewon Min**[♡]
Luke Zettlemoyer[♡] **Yejin Choi**[♡♠] **Hannaneh Hajishirzi**[♡♠]
[♡]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for Artificial Intelligence `liujc@cs.washington.edu`

n-grams Today

Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens

Jiacheng Liu[♡] **Sewon Min**[♡]
Luke Zettlemoyer[♡] **Yejin Choi**[♡♠] **Hannaneh Hajishirzi**[♡♠]
[♡]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for Artificial Intelligence liujc@cs.washington.edu

- Clever use of smoothing to create n-gram LMs where $n = \infty$, at least in principle

n-grams Today

Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens

Jiacheng Liu[♡] **Sewon Min**[♡]
Luke Zettlemoyer[♡] **Yejin Choi**[♡♠] **Hannaneh Hajishirzi**[♡♠]
[♡]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for Artificial Intelligence `liujc@cs.washington.edu`

- Clever use of smoothing to create n-gram LMs where $n = \infty$, at least in principle

  - Trained on several open text corpora: Dolma, RedPajama, Pile, and C4
    - Same corpora are used to train LLMs

n-grams Today

Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens

Jiacheng Liu[♡] **Sewon Min**[♡]
Luke Zettlemoyer[♡] **Yejin Choi**[♡♠] **Hannaneh Hajishirzi**[♡♠]
[♡]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for Artificial Intelligence liujc@cs.washington.edu

- Clever use of smoothing to create n-gram LMs where $n = \infty$, at least in principle

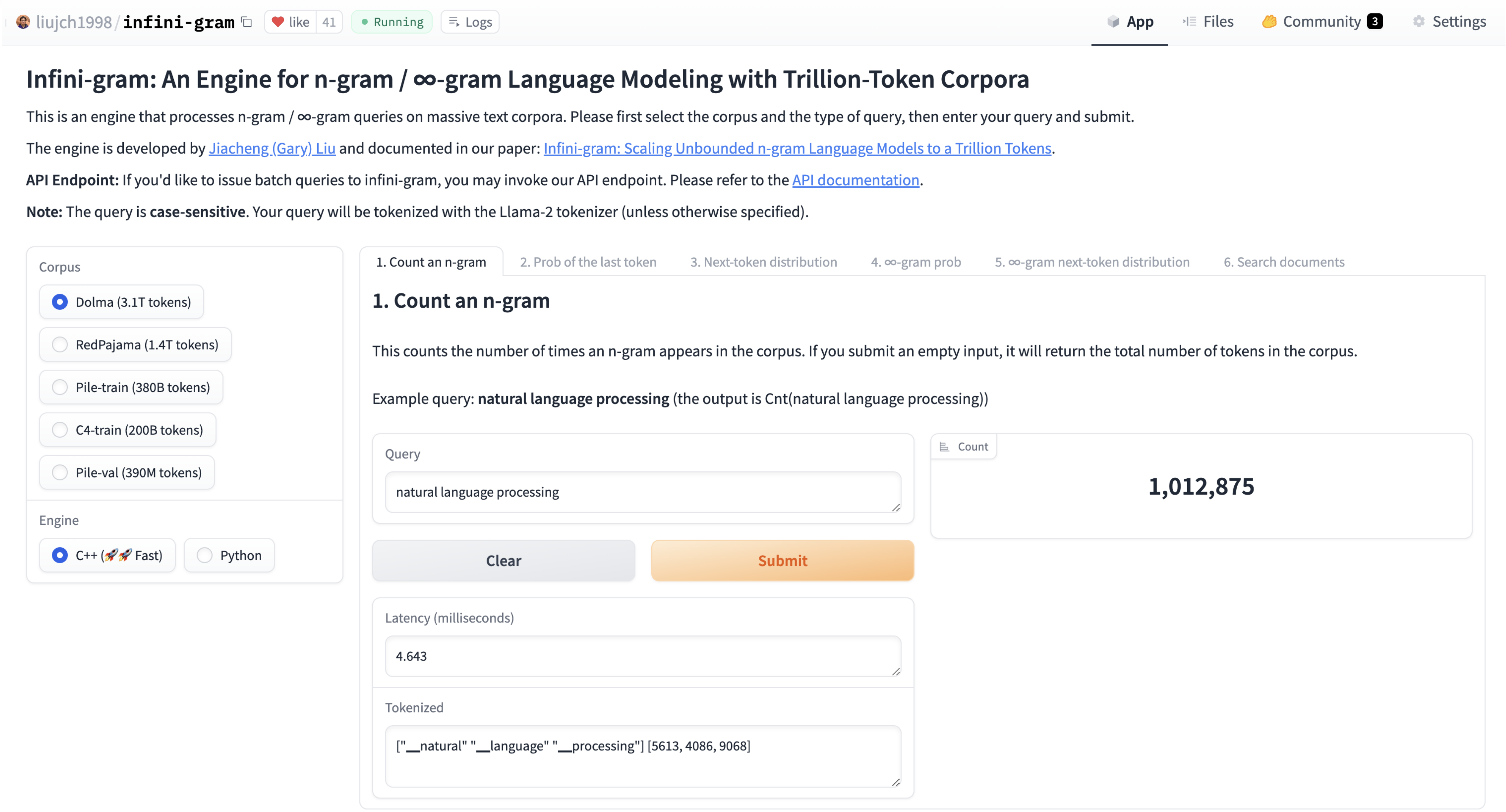
  - Trained on several open text corpora: Dolma, RedPajama, Pile, and C4
    - Same corpora are used to train LLMs
  - Several applications: search for n-grams, their counts and their probabilities, use them for generation, etc.

USC Viterbi

n-grams Today

Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens

Jiacheng Liu[♡] **Sewon Min**[♡]
Luke Zettlemoyer[♡] **Yejin Choi**[♡♠] **Hannaneh Hajishirzi**[♡♠]
[♡]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for Artificial Intelligence liujc@cs.washington.edu



- Clever use of smoothing to create n-gram LMs where $n = \infty$, at least in principle
- Trained on several open text corpora: Dolma, RedPajama, Pile, and C4
  - Same corpora are used to train LLMs
- Several applications: search for n-grams, their counts and their probabilities, use them for generation, etc.

Lecture Outline

- Announcements
- Recap
  - n-gram Language Models
  - Zeros!
- Smoothing
- Basics of Supervised Machine Learning
  I. Data: Preprocessing and Feature Extraction
  II. Model:
      I. Logistic Regression
  III. Loss
  IV. Optimization Algorithm
  V. Inference

Basics of Supervised Machine Learning

Ingredients of Supervised Machine Learning

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \dots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \dots, x_d]$,

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
     - e.g. word embeddings

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
  - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
    - e.g. word embeddings

II. Model

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \dots N\}$
  - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \dots, x_d]$,
    - e.g. word embeddings

II. **Model**
  - A classification function that computes $\hat{y}$, the estimated class, via $p(y \mid x)$

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y \mid x)$
     - e.g. logistic regression, naïve Bayes

USC Viterbi

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y \mid x)$
     - e.g. logistic regression, naïve Bayes

III. **Loss**

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
    - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
        - e.g. word embeddings

II. **Model**
    - A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
        - e.g. logistic regression, naïve Bayes

III. **Loss**
    - An objective function for learning

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1...N\}$
- $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, ..., x_d]$,
  - e.g. word embeddings

II. Model
- A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
  - e.g. logistic regression, naïve Bayes

III. Loss
- An objective function for learning
  - e.g. cross-entropy loss, L_{CE}

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \dots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \dots, x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y \mid x)$
     - e.g. logistic regression, naïve Bayes

III. **Loss**
   - An objective function for learning
     - e.g. cross-entropy loss, L_{CE}

IV. **Optimization**

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \dots N\}$
    - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \dots, x_d]$,
        - e.g. word embeddings

II. **Model**
    - A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
        - e.g. logistic regression, naïve Bayes

III. **Loss**
    - An objective function for learning
        - e.g. cross-entropy loss, L_{CE}

IV. **Optimization**
    - An algorithm for optimizing the objective function

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \dots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \dots, x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
     - e.g. logistic regression, naïve Bayes

III. **Loss**
   - An objective function for learning
     - e.g. cross-entropy loss, L_{CE}

IV. **Optimization**
   - An algorithm for optimizing the objective function
     - e.g. stochastic gradient descent

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1...N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
     - e.g. logistic regression, naïve Bayes

III. **Loss**
   - An objective function for learning
     - e.g. cross-entropy loss, L_{CE}

IV. **Optimization**
   - An algorithm for optimizing the objective function
     - e.g. stochastic gradient descent

V. **Inference** / Evaluation

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
    - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
        - e.g. word embeddings

II. **Model**
    - A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
        - e.g. logistic regression, naïve Bayes

III. **Loss**
    - An objective function for learning
        - e.g. cross-entropy loss, L_{CE}

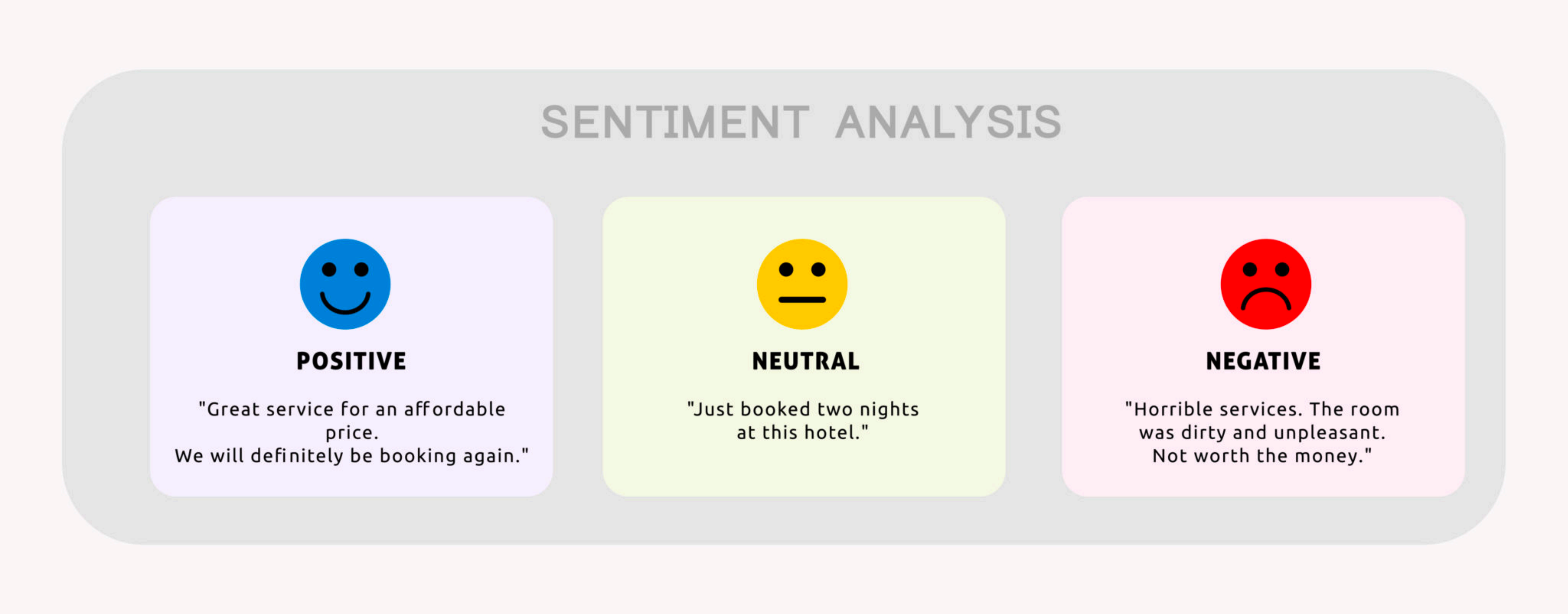
IV. **Optimization**
    - An algorithm for optimizing the objective function
        - e.g. stochastic gradient descent

V. **Inference** / Evaluation

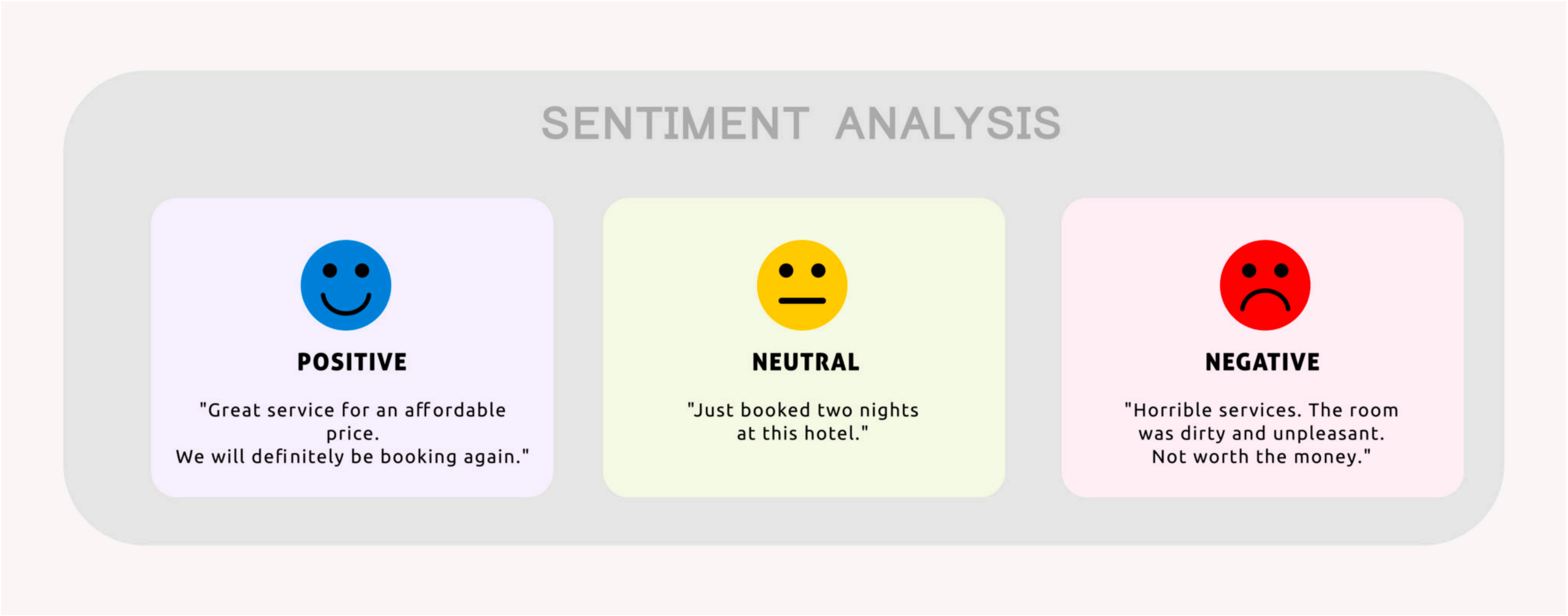
Learning Phase

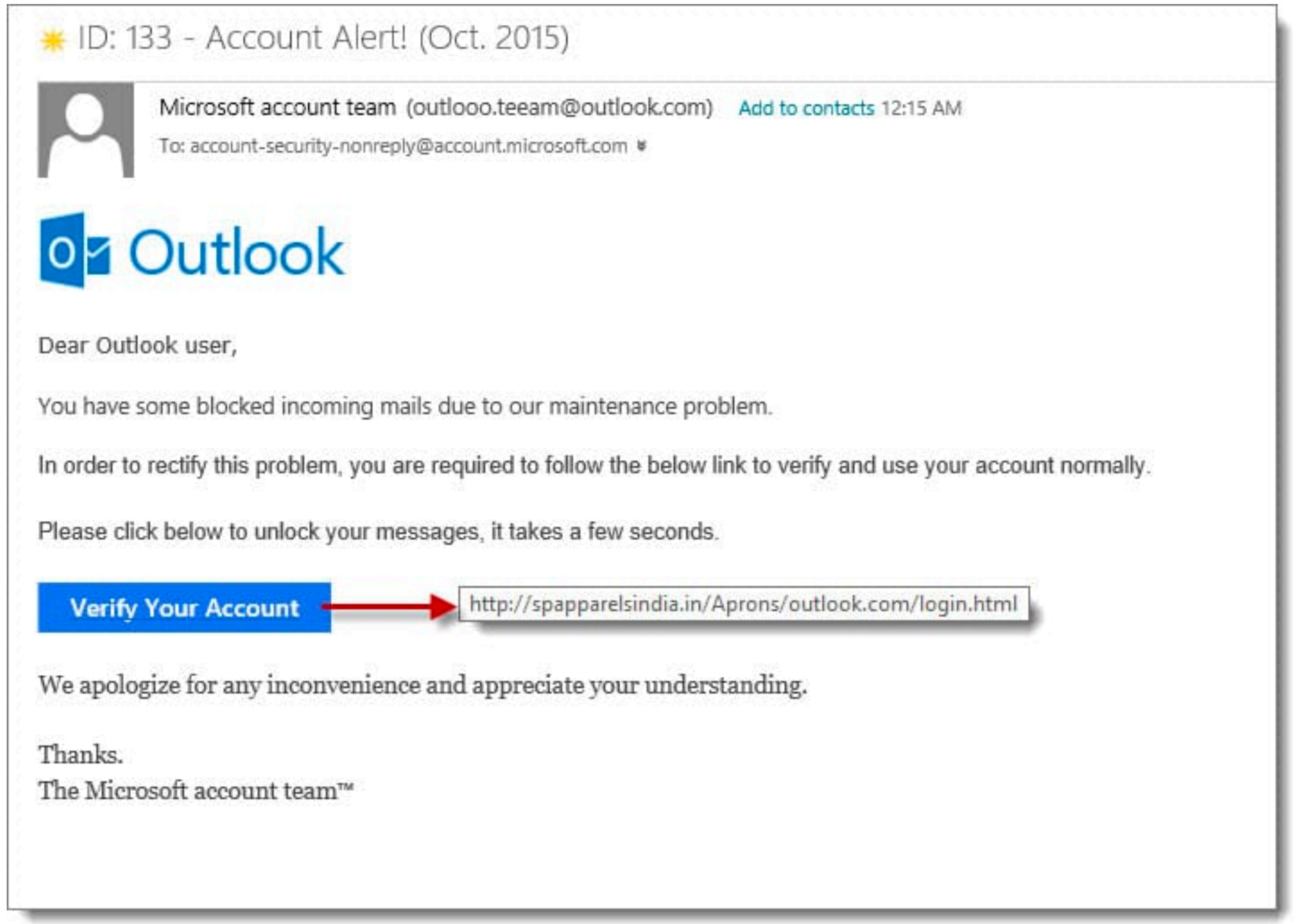
Text Classification Tasks

Text Classification Tasks

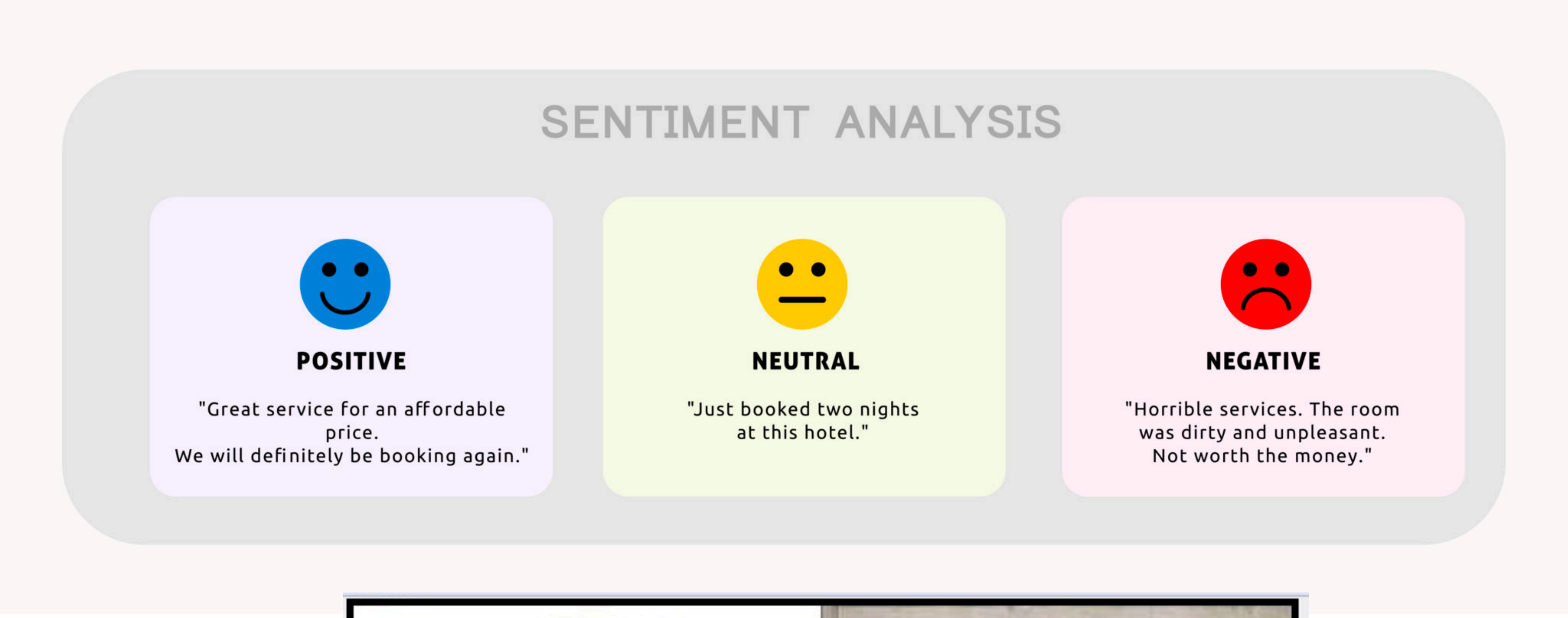


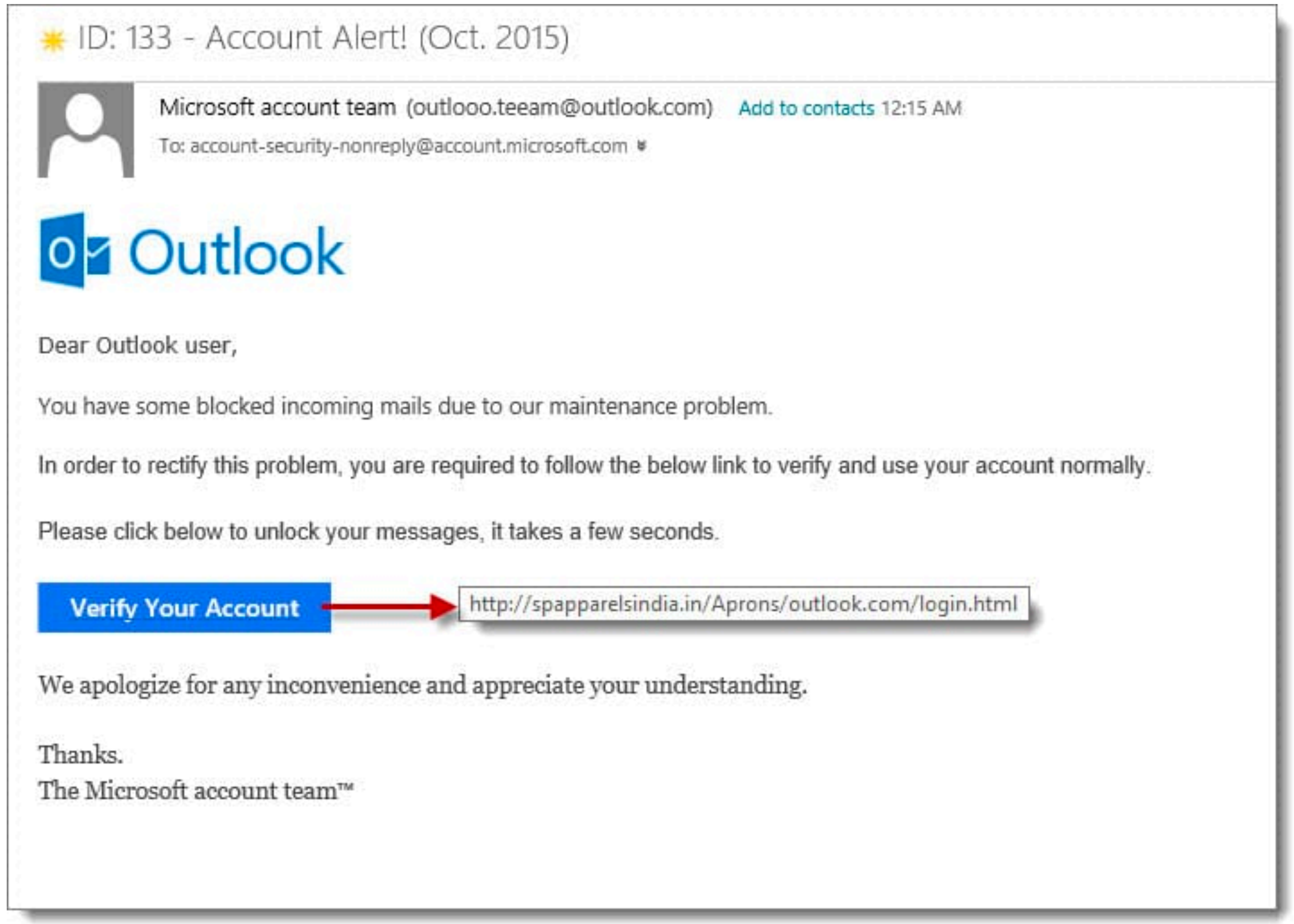
Text Classification Tasks



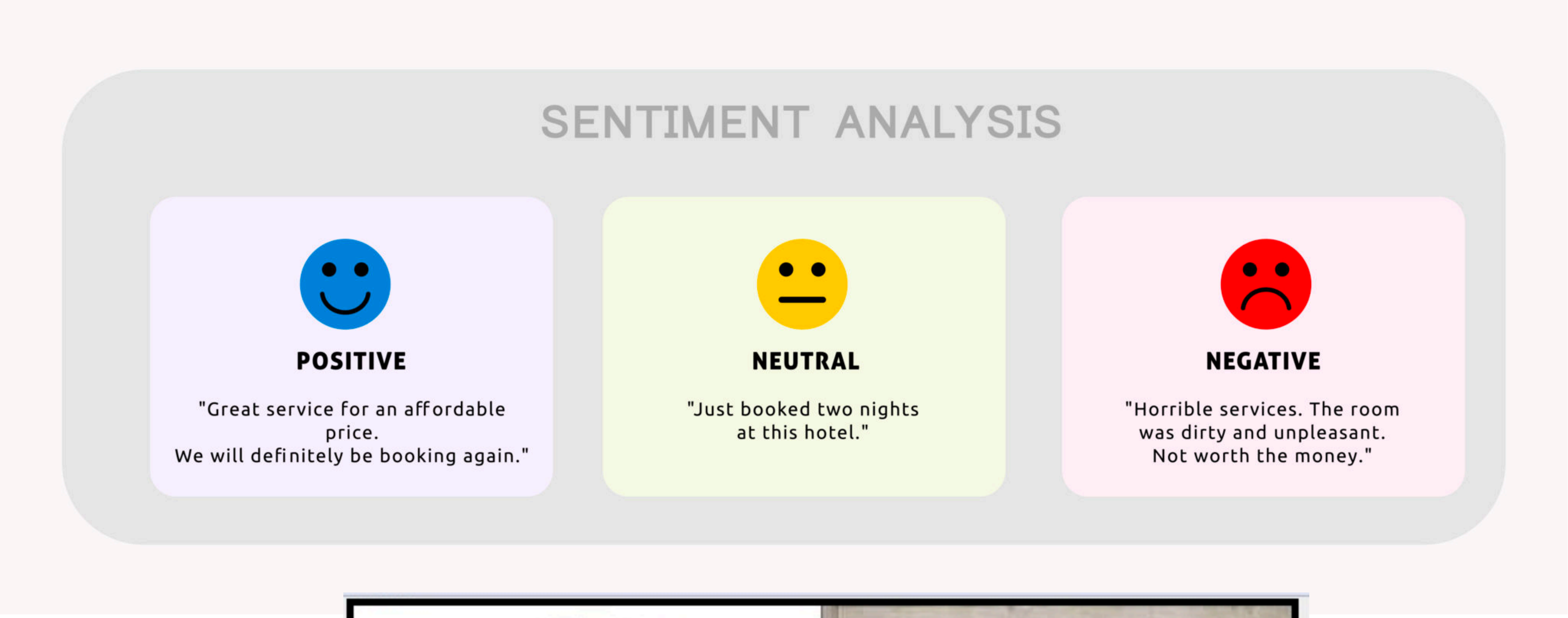


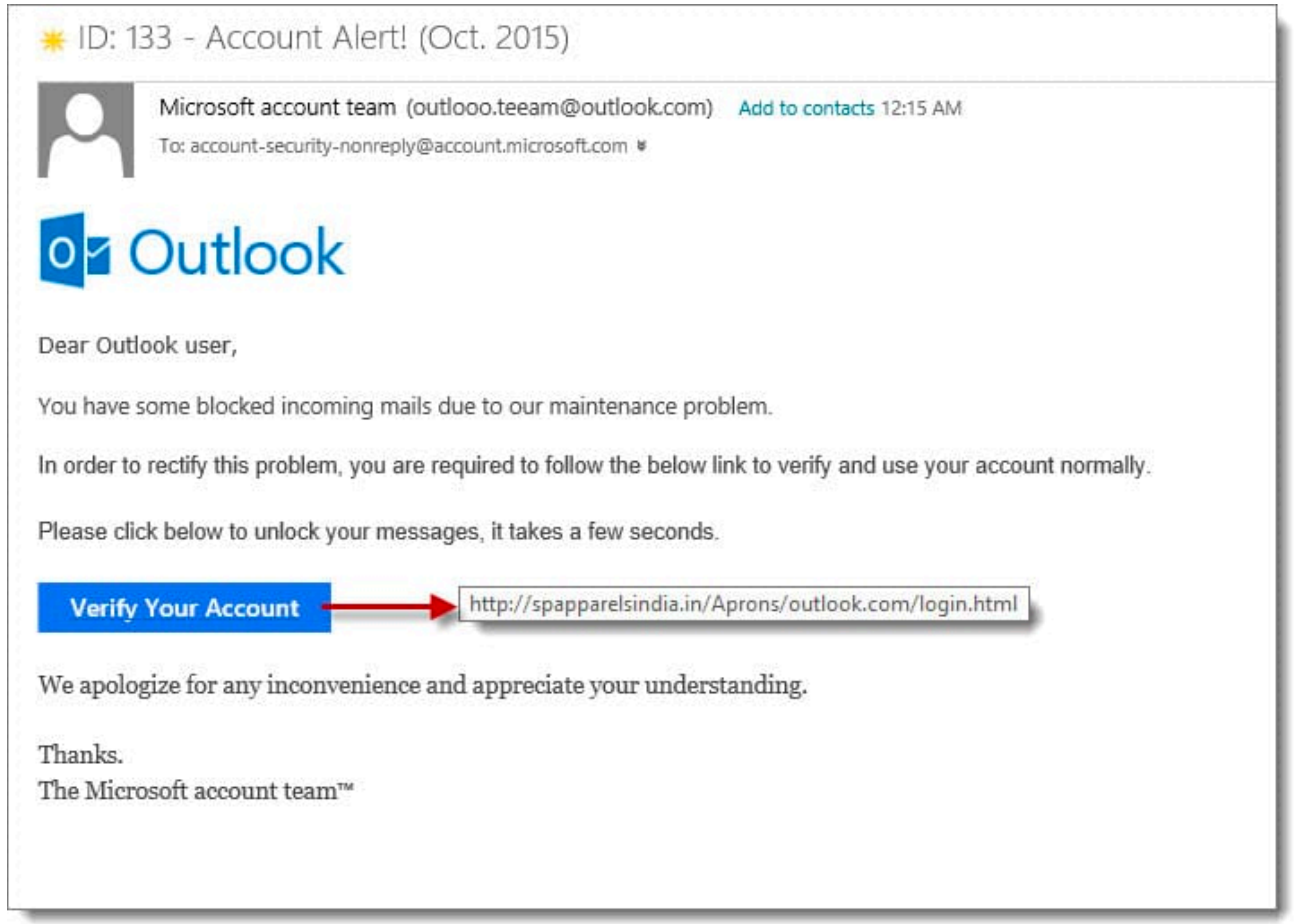
Text Classification Tasks





Text Classification Tasks





Not just NLP, classification is a general ML technique often applied across a wide variety of prediction tasks!

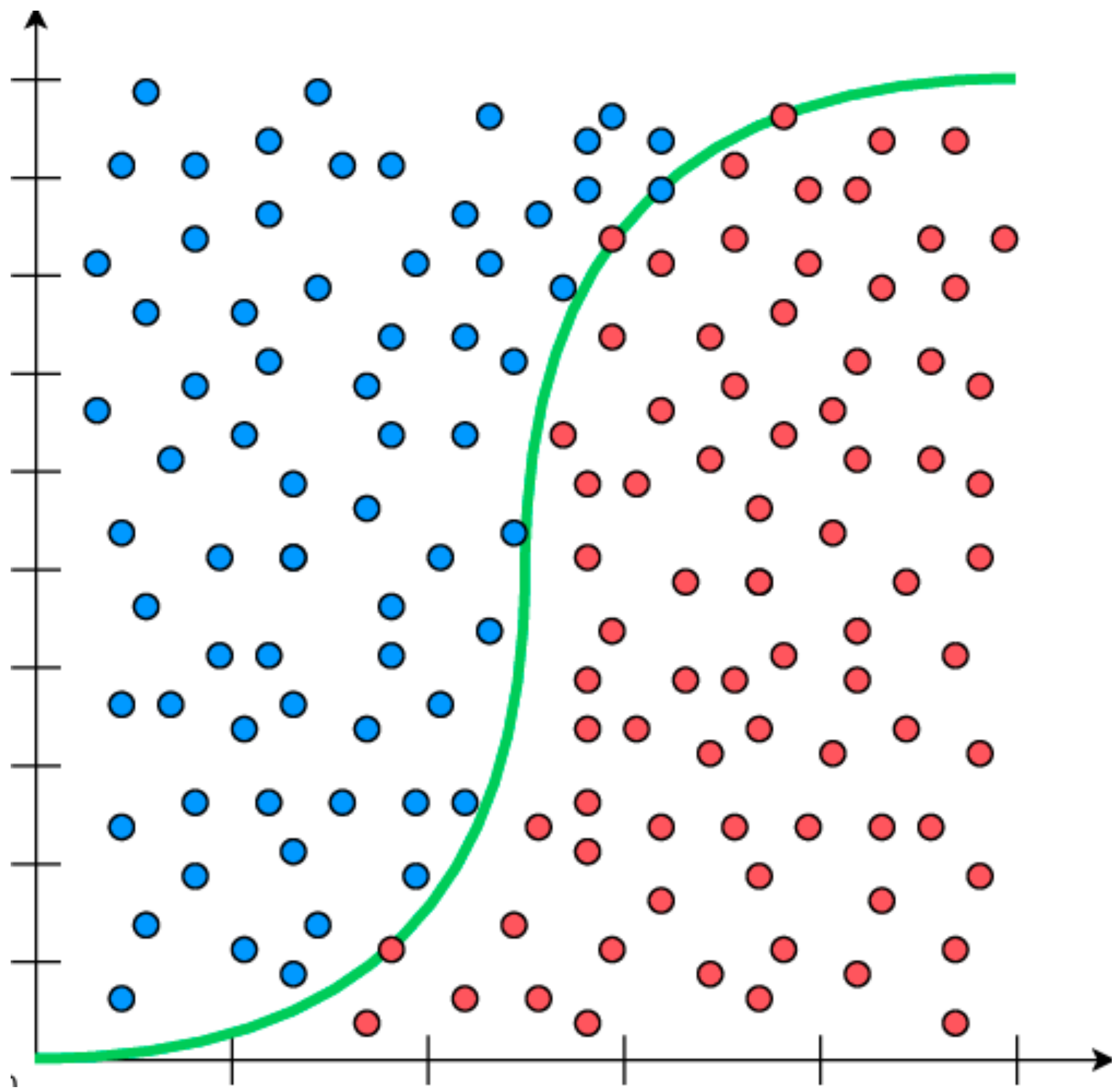
Text Classification Setup

Text Classification Setup

- Input:
  - a document x
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$

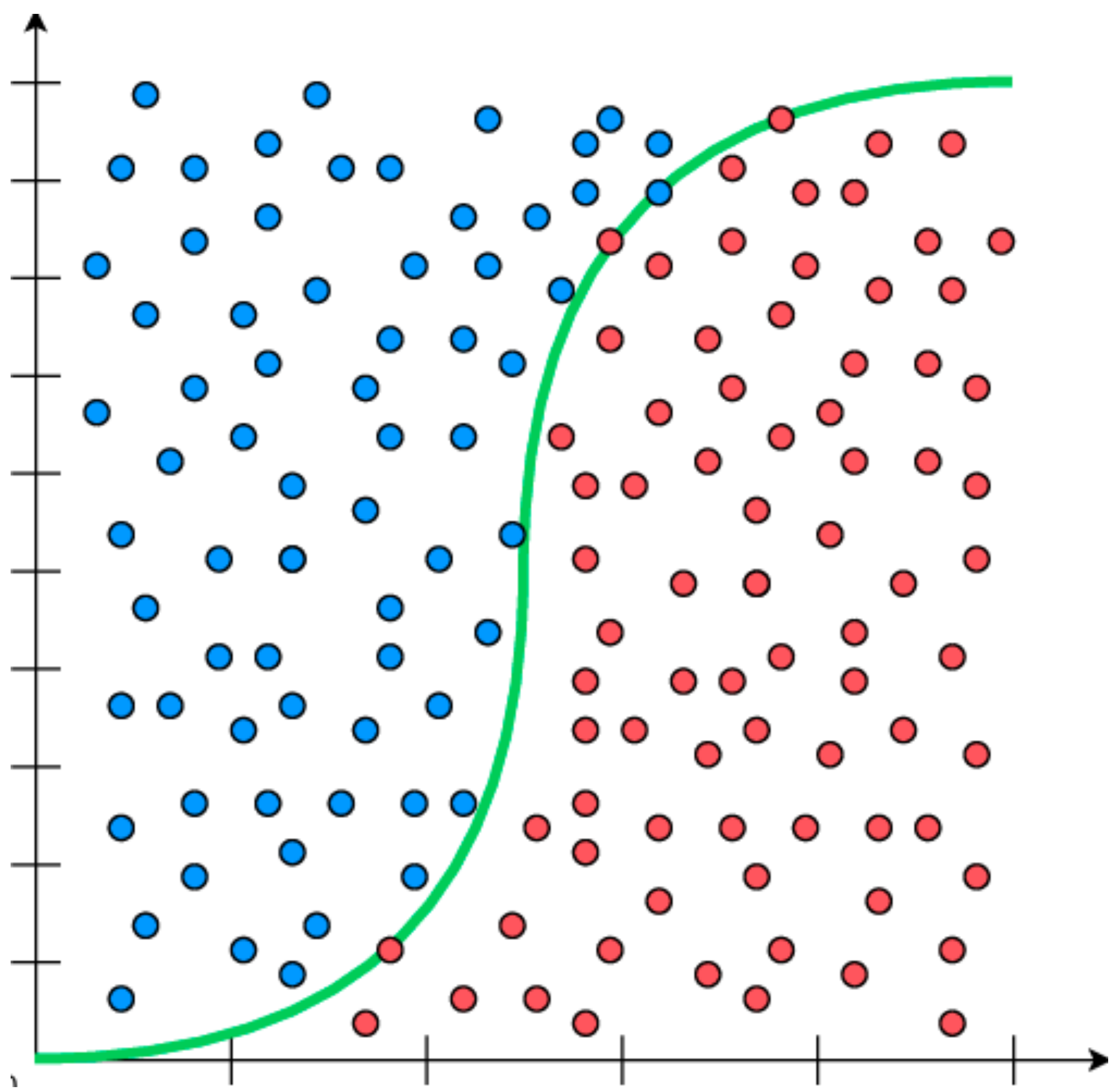
Text Classification Setup

- Input:
  - a document x
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$



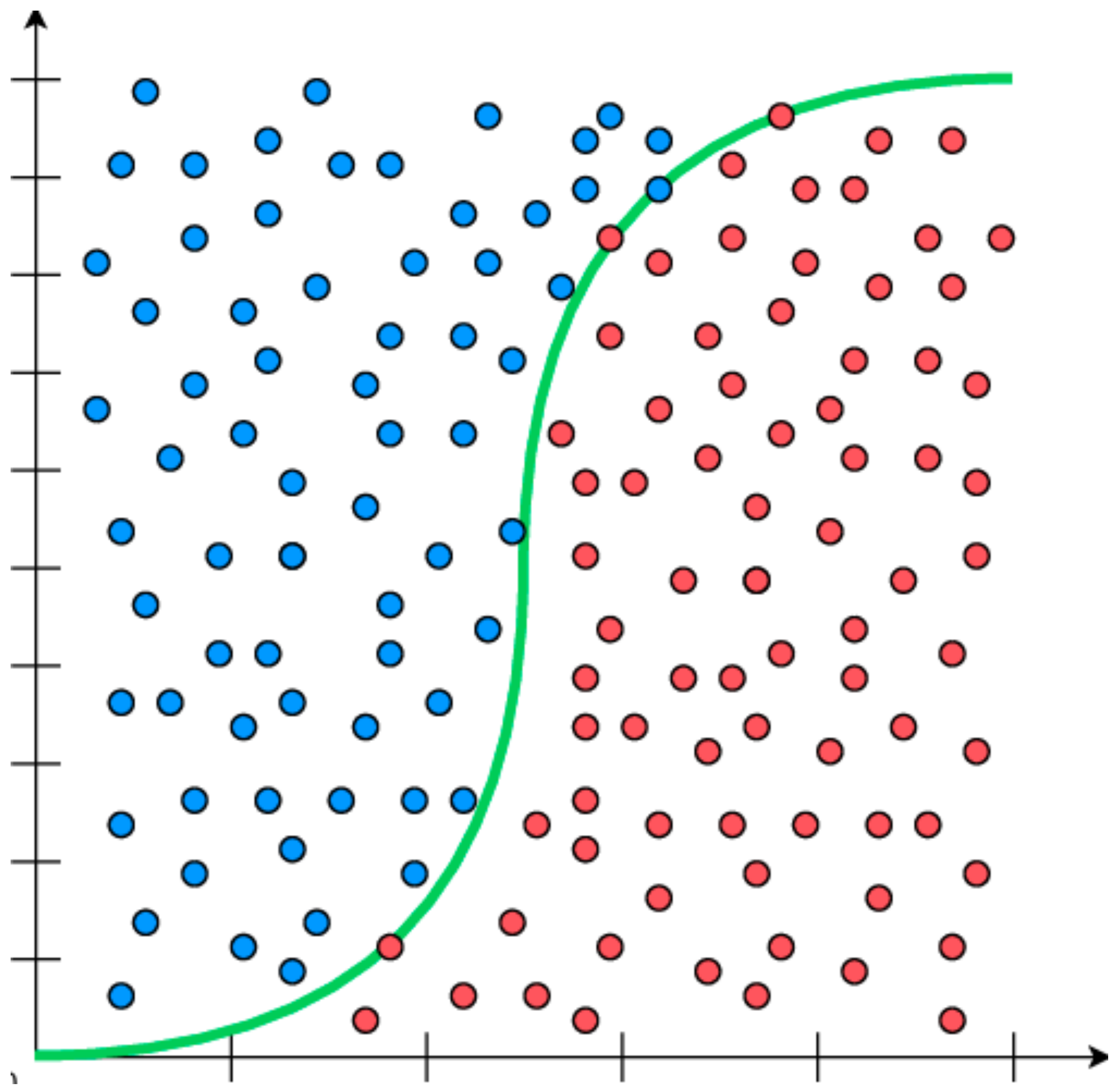
Text Classification Setup

- Input:
  - a document x
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$
  - a label y from a fixed set of classes $C = c_1, c_2, \ldots, c_J$



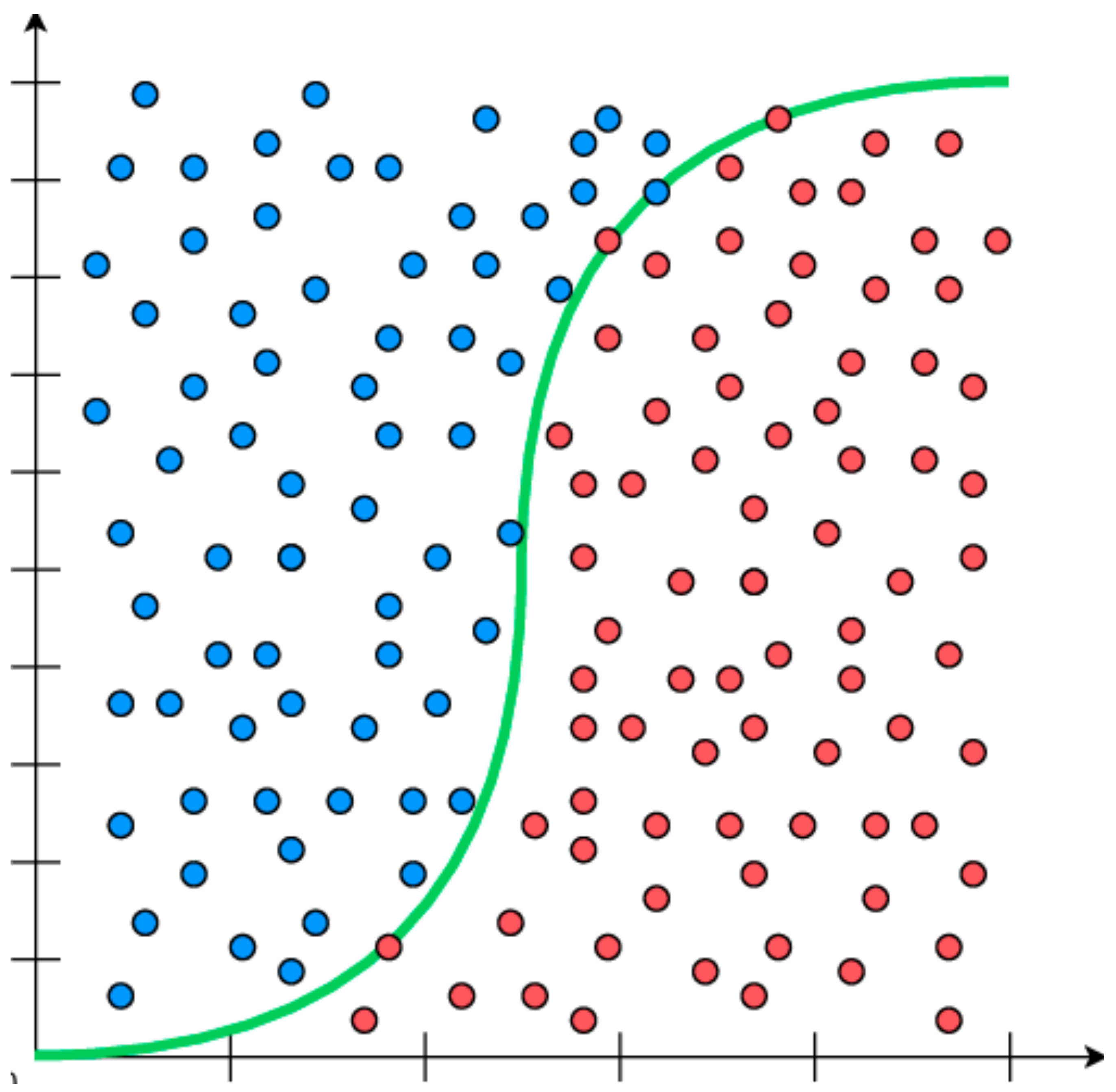
Text Classification Setup

- Input:
  - a document x
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}]$
  - a label y from a fixed set of classes $C = c_1, c_2, \dots, c_J$
- Output: a predicted class $\hat{y} \in C$



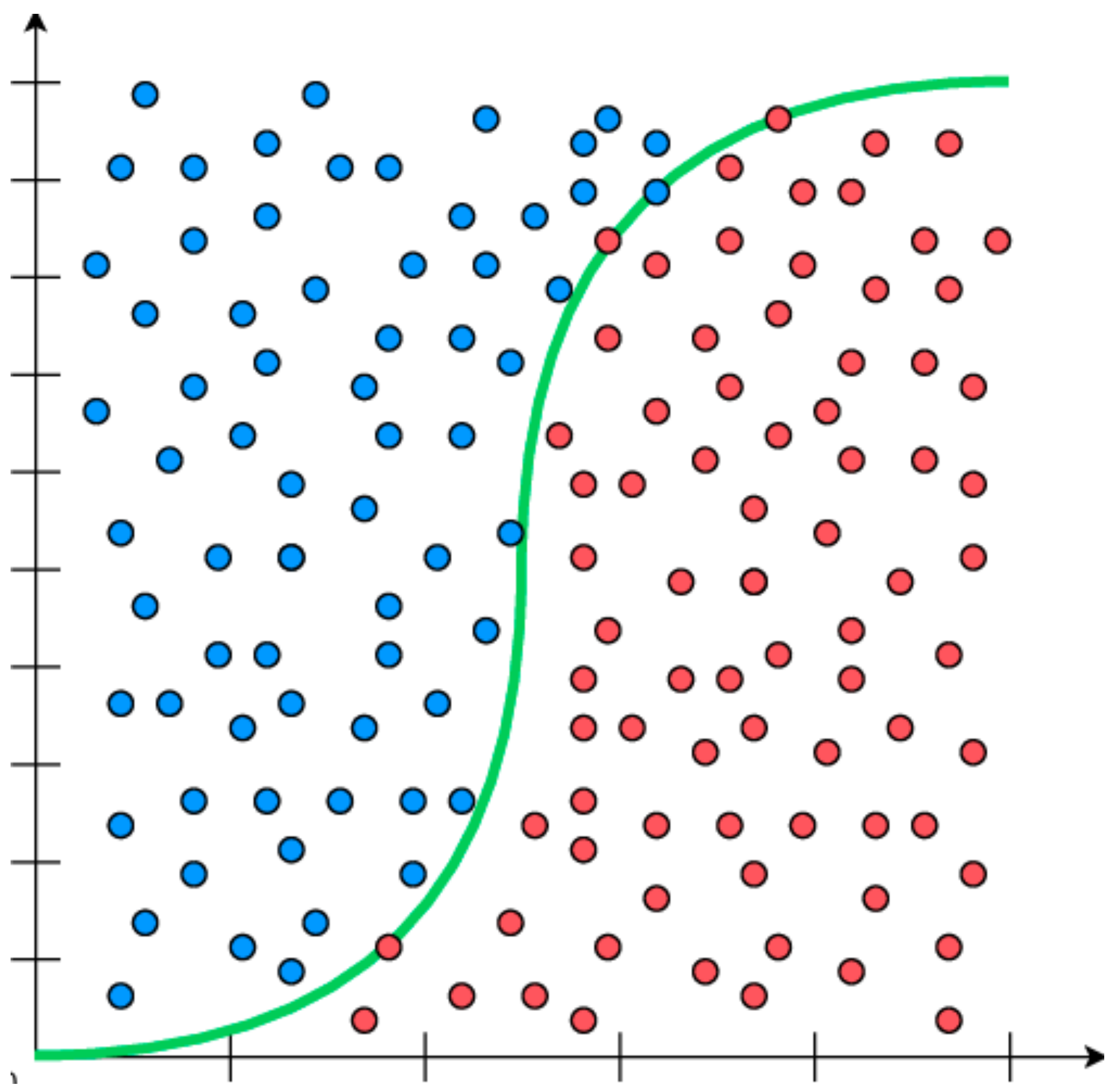
Text Classification Setup

- Input:
  - a document x
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$
  - a label y from a fixed set of classes $C = c_1, c_2, \ldots, c_J$
- Output: a predicted class $\hat{y} \in C$
- Setting for Binary Classification: given a series of input / output pairs:
  - $(x^{(i)}, y^{(i)})$ where label $y^{(i)} \in C = \{0,1\}$



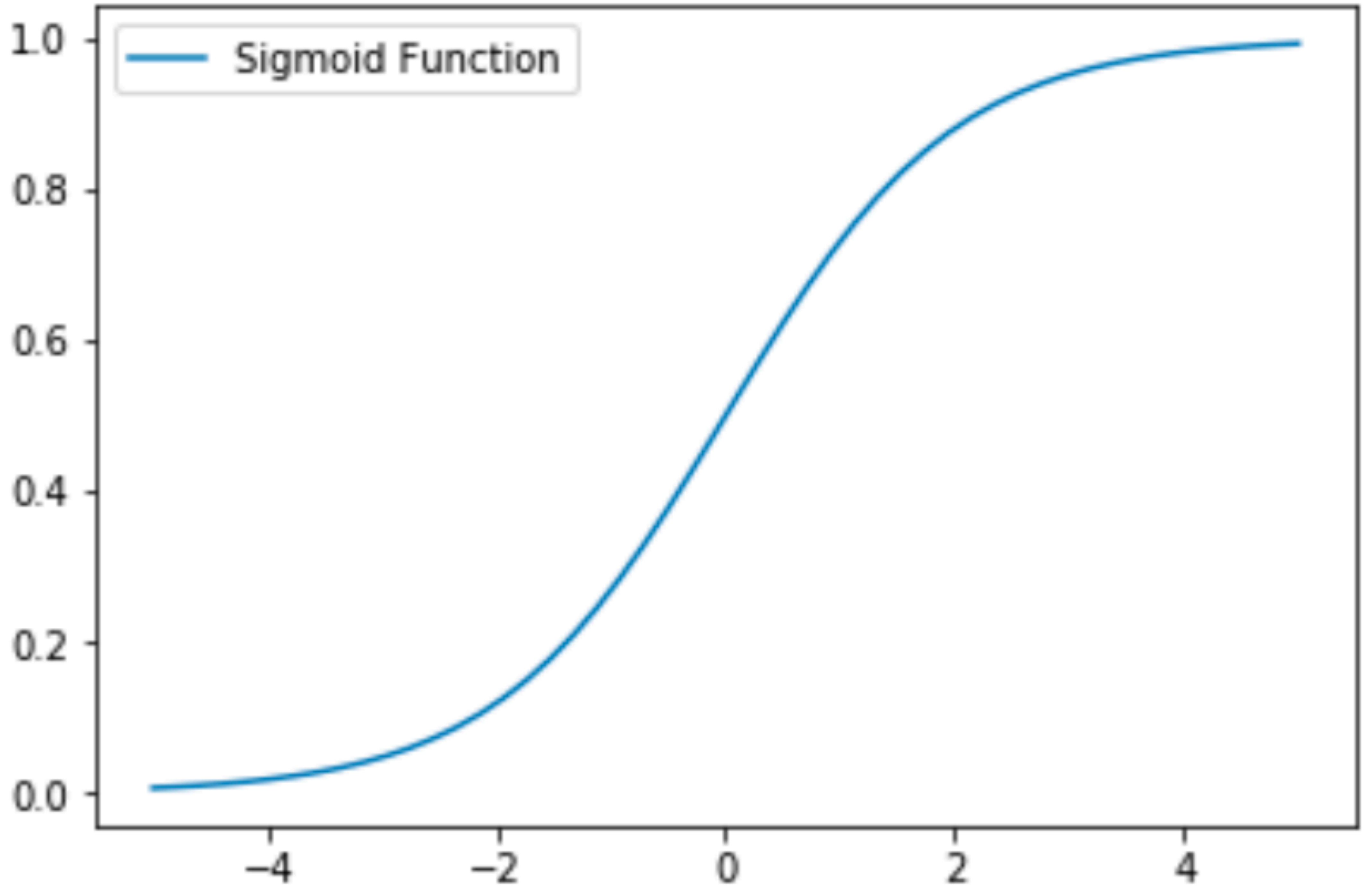
Text Classification Setup

- Input:
  - a document x
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$
  - a label y from a fixed set of classes $C = c_1, c_2, \ldots, c_J$
- Output: a predicted class $\hat{y} \in C$
- Setting for Binary Classification: given a series of input / output pairs:
  - $(x^{(i)}, y^{(i)})$ where label $y^{(i)} \in C = \{0,1\}$
- Goal of Binary Classification
  - At test time, for input x^{test}, compute an output: a predicted class $\hat{y}^{test} \in \{0,1\}$



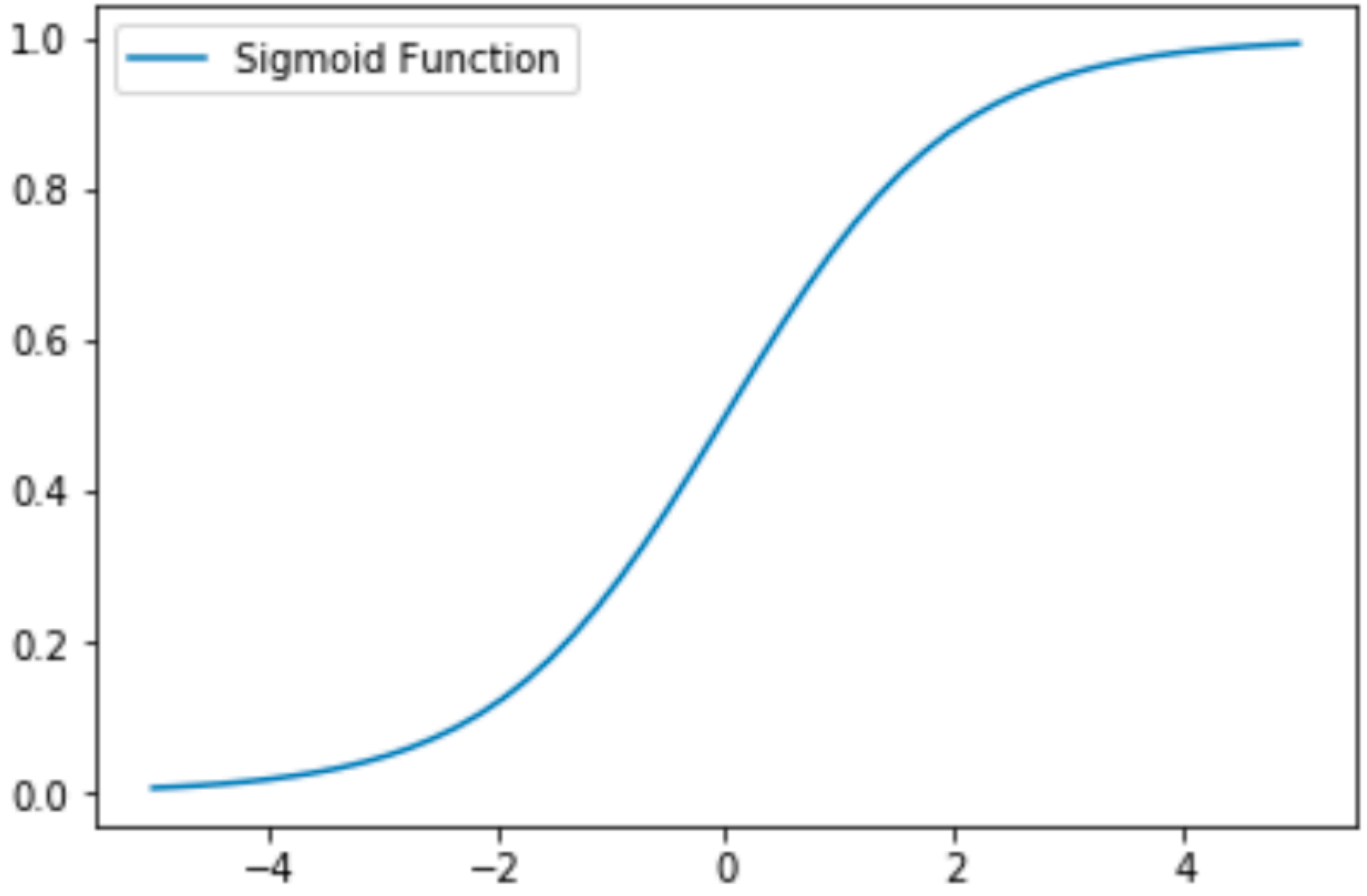
Example: Logistic Regression

Example: Logistic Regression



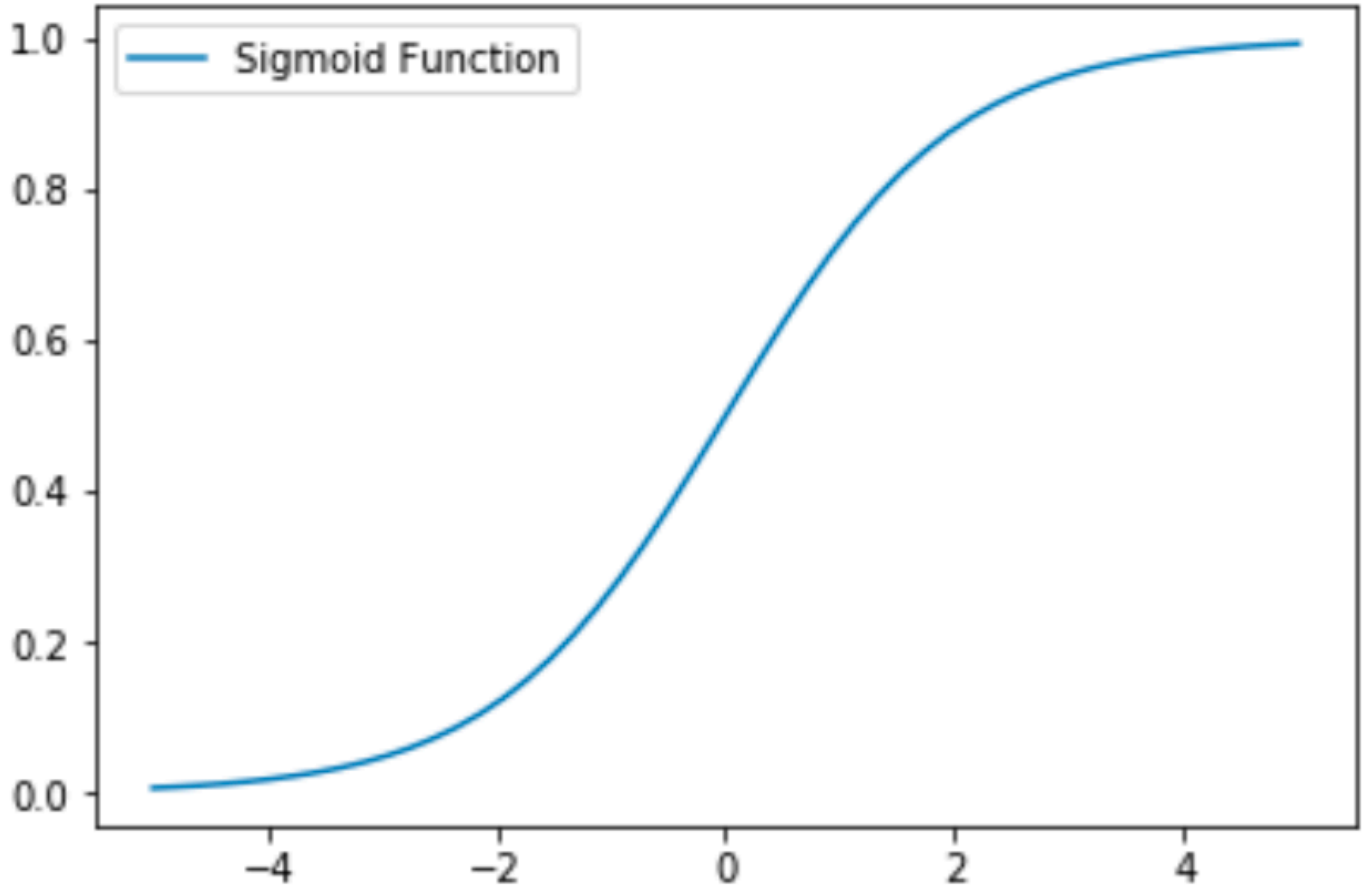
Example: Logistic Regression

- Important analytic tool in natural and social sciences



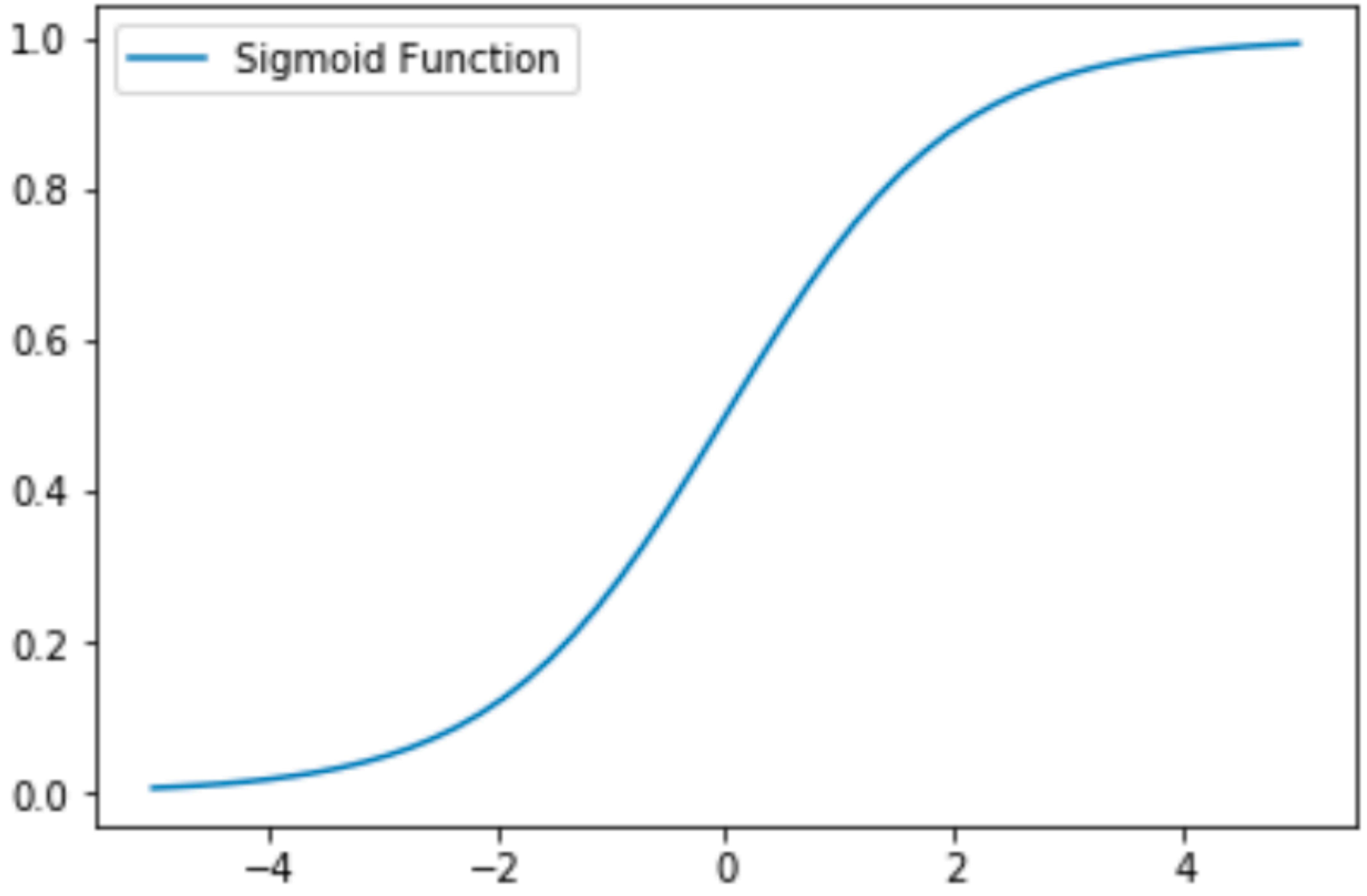
Example: Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification



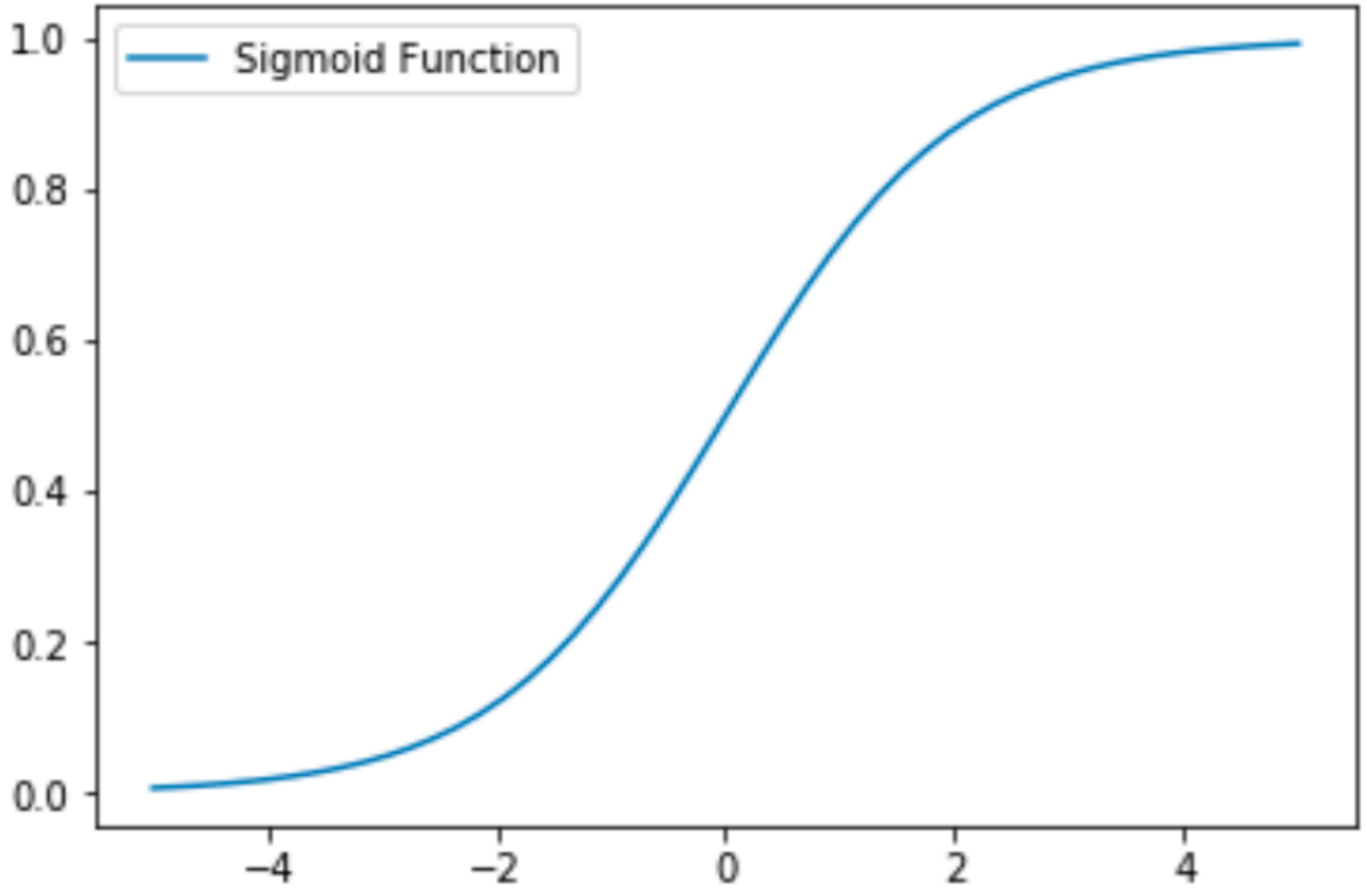
Example: Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks



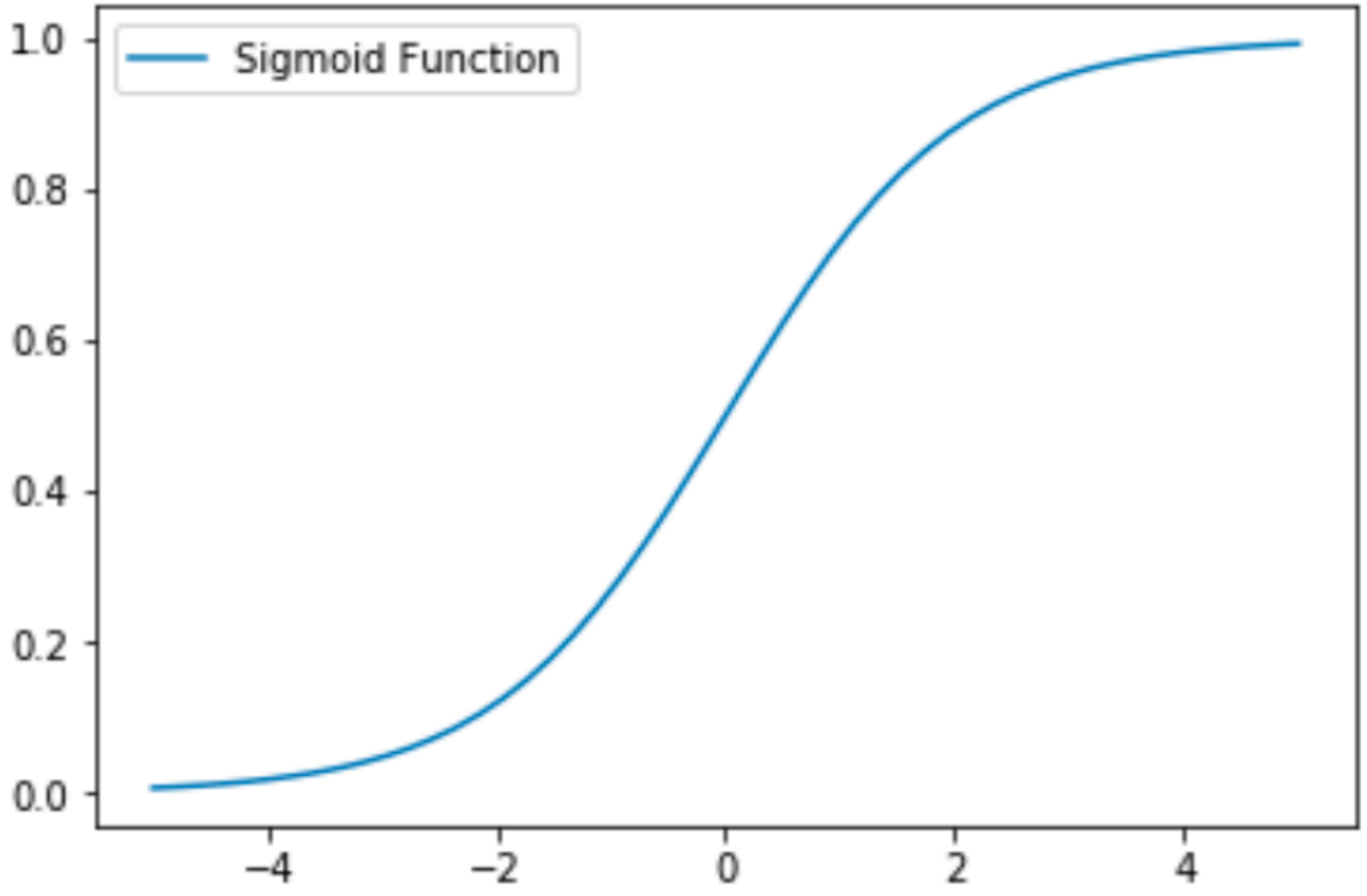
Example: Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks
- Logistic regression is a discriminative classifier
  - Learn a model that can (given the input) distinguish between different classes



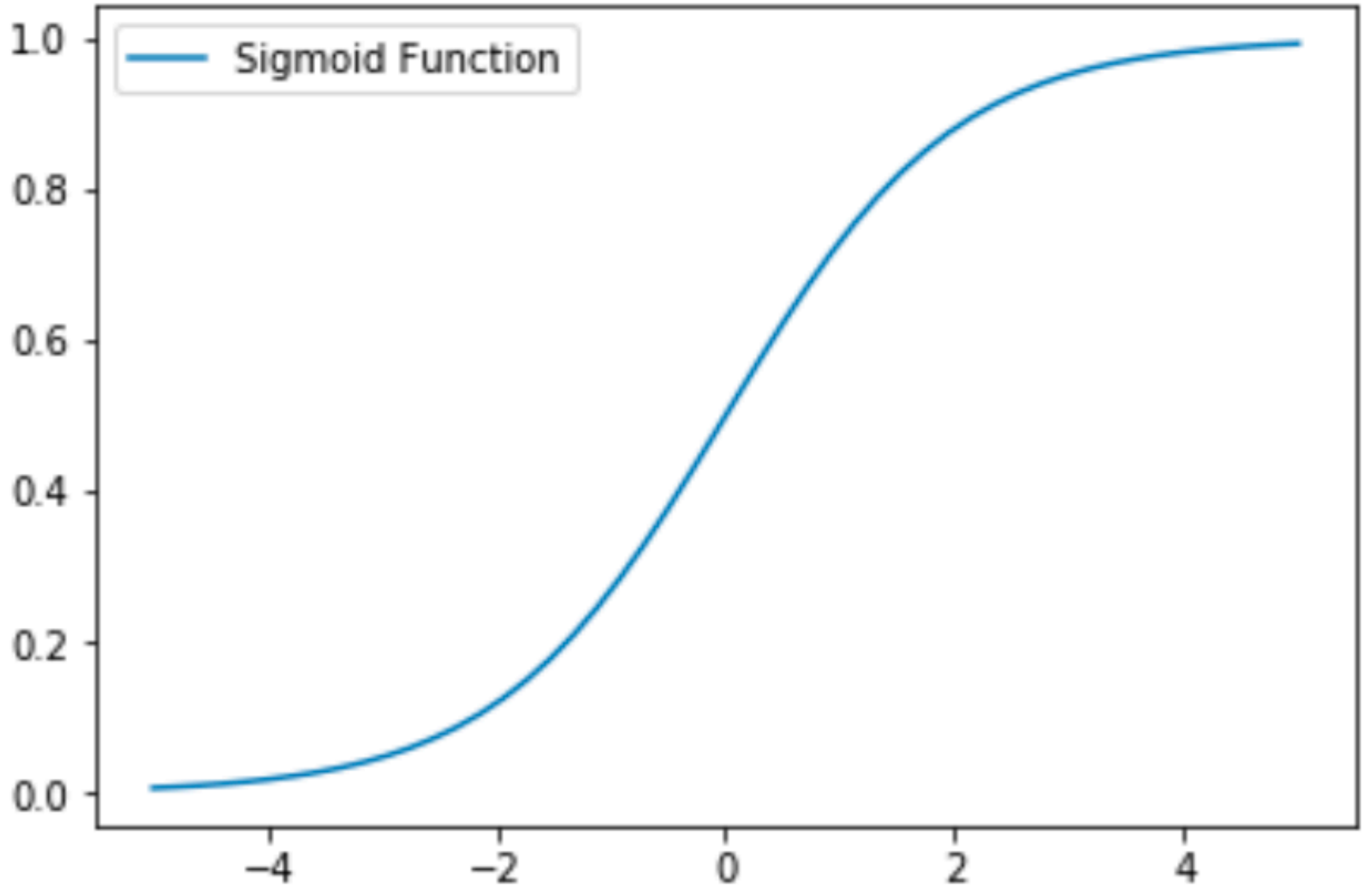
Example: Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks
- Logistic regression is a discriminative classifier
  - Learn a model that can (given the input) distinguish between different classes
- Other classification algorithms: Naïve Bayes, K-Nearest Neighbors, Decision Trees, SVMs



Example: Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks
- Logistic regression is a discriminative classifier
    - Learn a model that can (given the input) distinguish between different classes
- Other classification algorithms: Naïve Bayes, K-Nearest Neighbors, Decision Trees, SVMs



Is language modeling a classification task?

Classification: Single Observation

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$
    - Sometimes we call the weights $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$

Parametric Model

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class

Parametric Model

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class
  - Binary logistic regression $\hat{y} \in \{0,1\}$

Parametric Model

Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class
  - Binary logistic regression $\hat{y} \in \{0,1\}$
  - Multinomial logistic regression (e.g. 5 classes): $\hat{y} \in \{0,1,2,3,4\}$

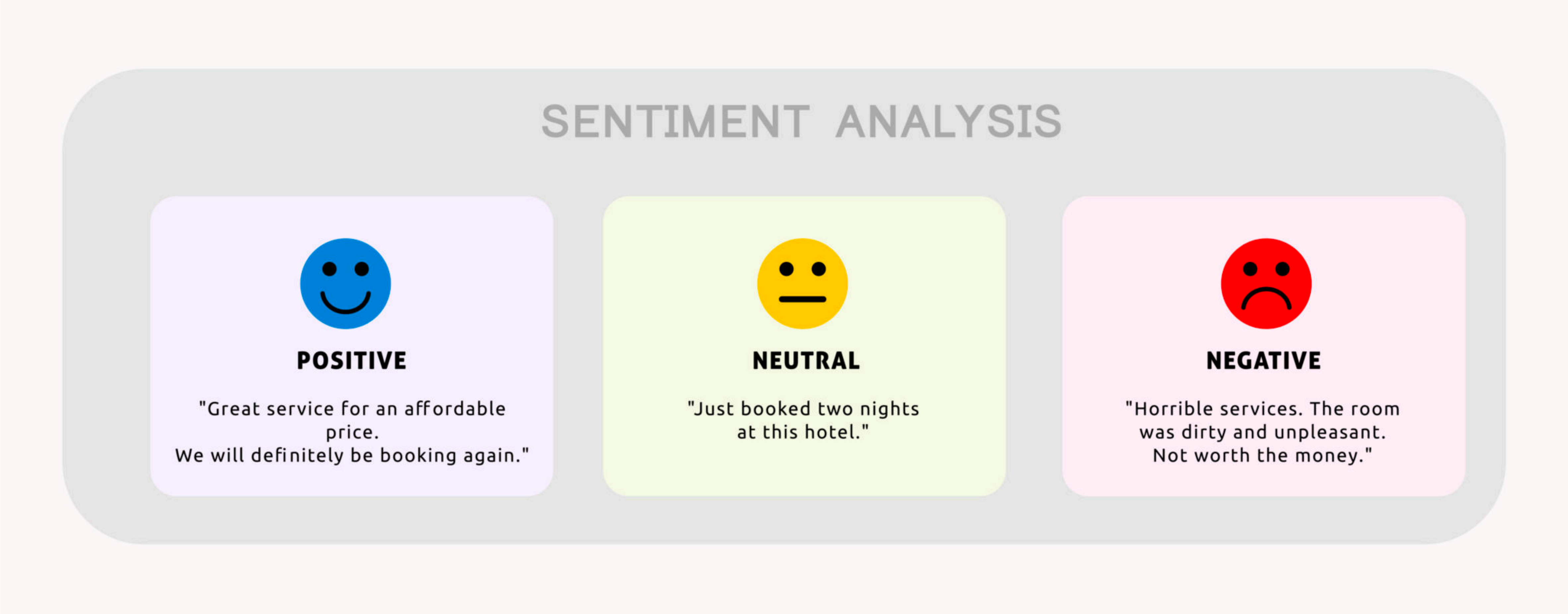
Parametric Model

Lecture Outline

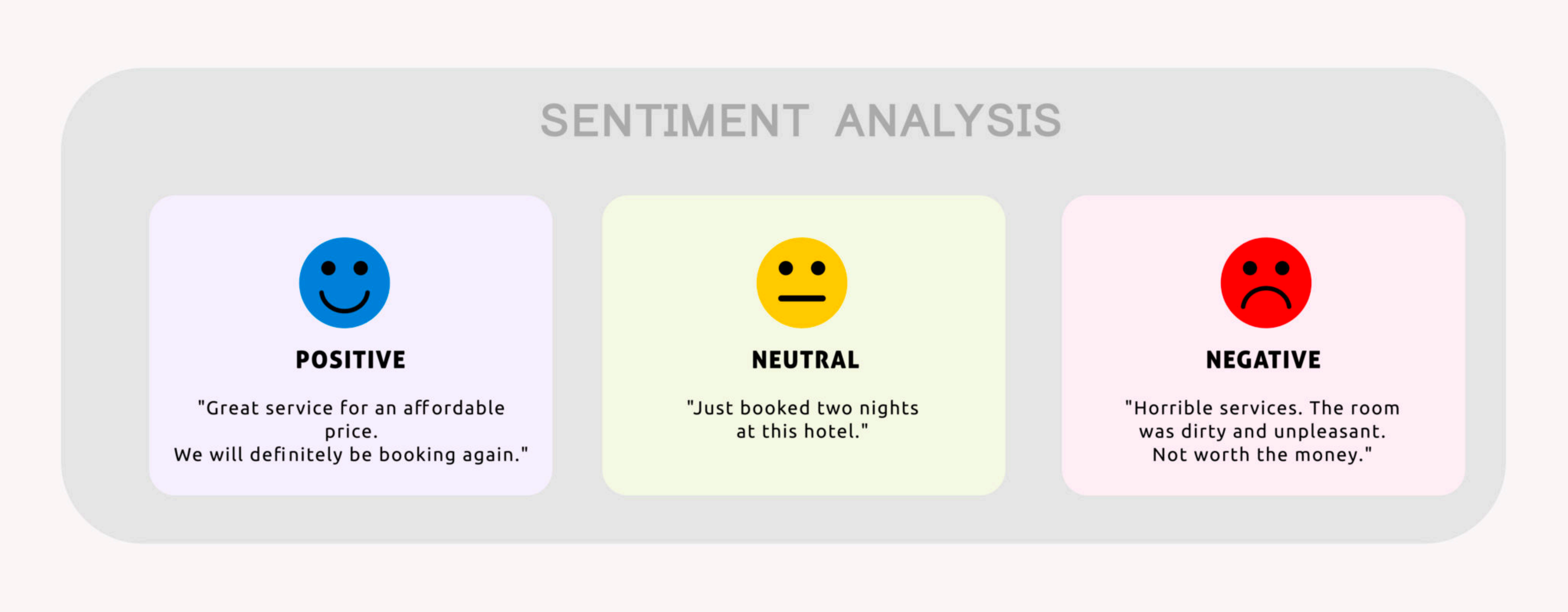
- Announcements
- Recap
  - n-gram Language Models
  - Zeros!
- Smoothing
- Basics of Supervised Machine Learning
  I. Data: Preprocessing and Feature Extraction
  II. Model:
     I. Logistic Regression
  III. Loss
  IV. Optimization Algorithm
  V. Inference

I. Data:
Preprocessing and Feature Extraction

Features in Classification

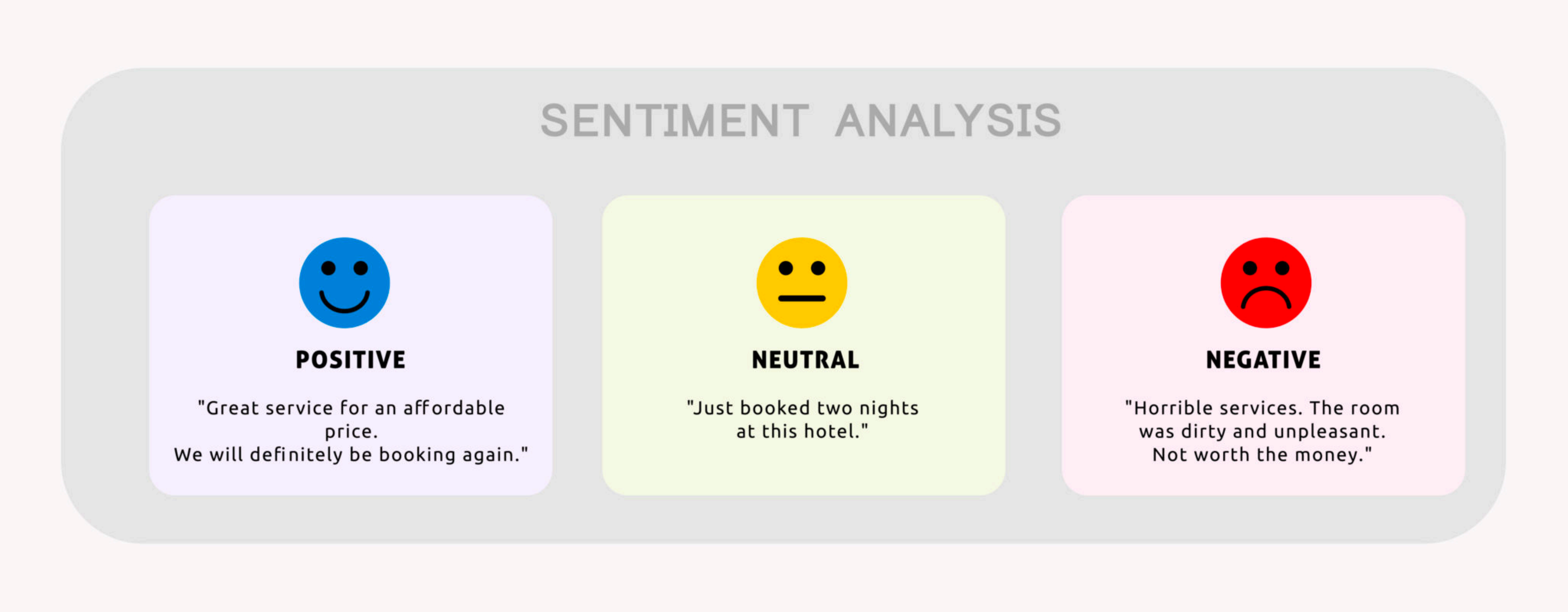


Features in Classification



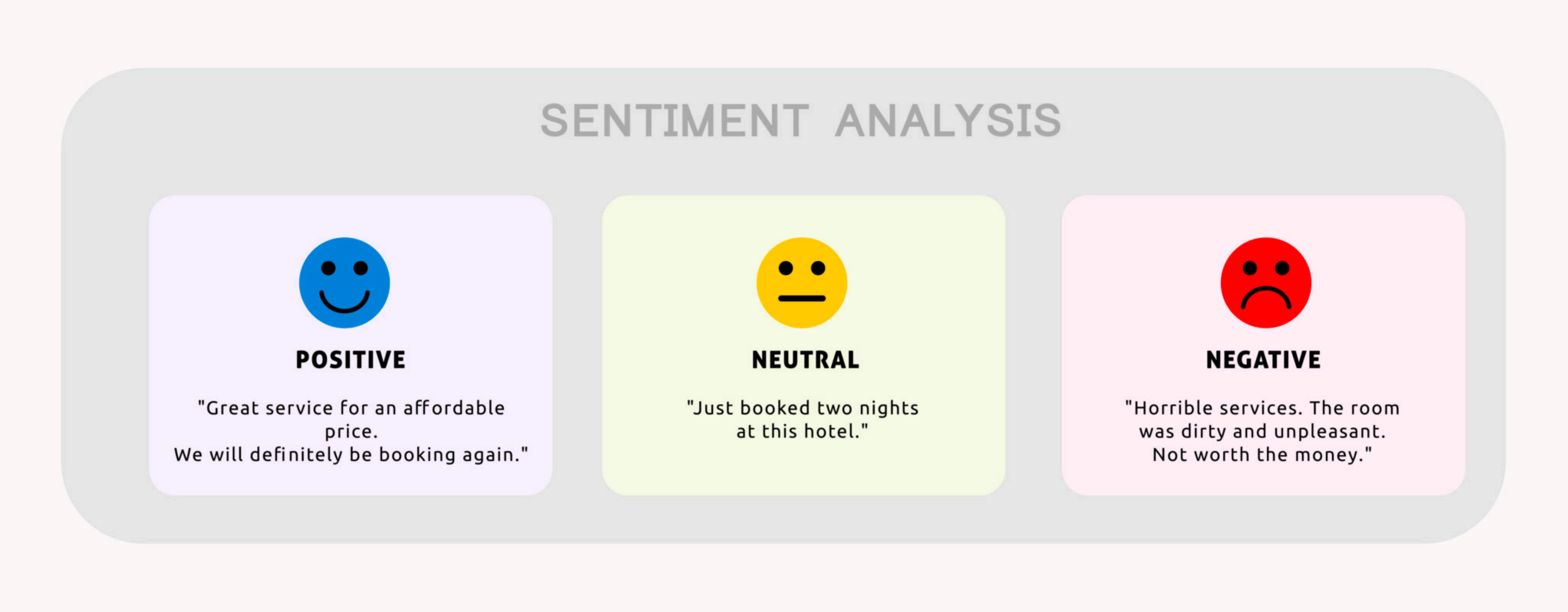
- Examples of feature x_i
  - x_i = "review contains 'awesome'"; $w_i = +10$
  - x_j = "review contains 'abysmal'"; $w_j = -10$
  - x_k = "review contains 'mediocre'"; $w_k = -2$

Features in Classification



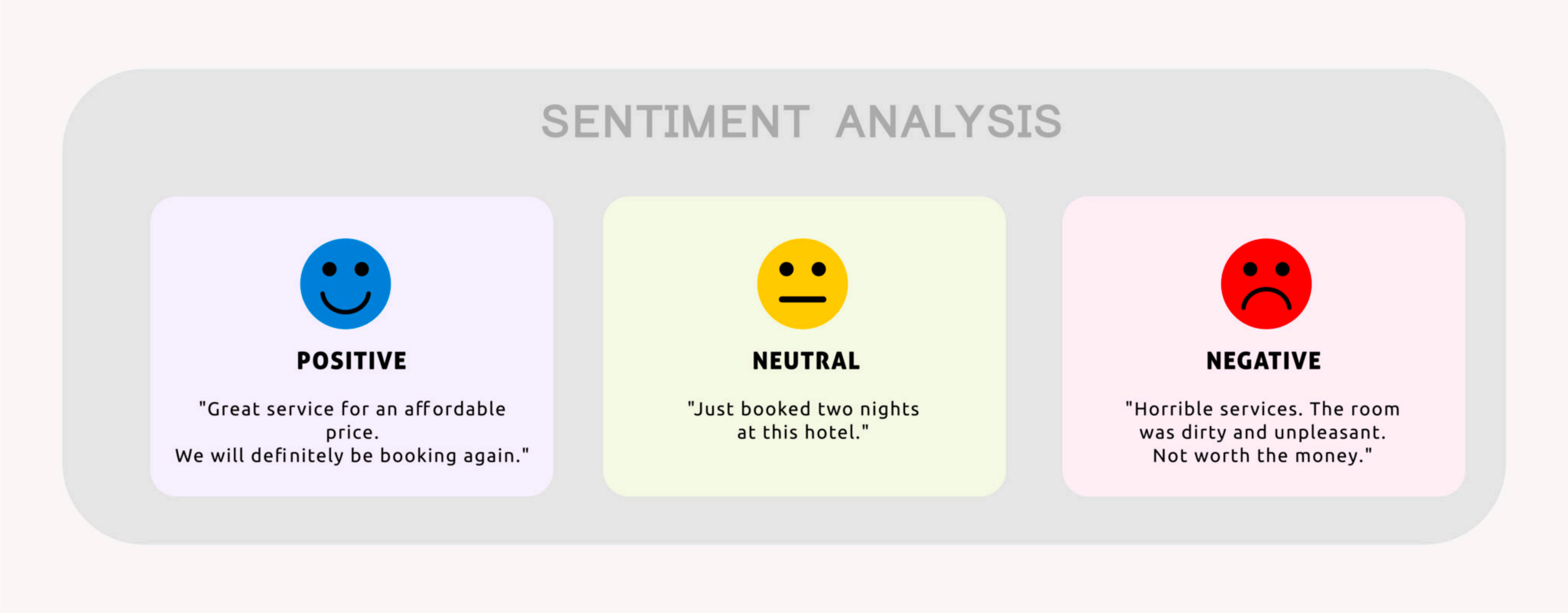
- Examples of feature x_i
  - $x_i =$ "review contains 'awesome'"; $w_i = +10$
  - $x_j =$ "review contains 'abysmal'"; $w_j = -10$
  - $x_k =$ "review contains 'mediocre'"; $w_k = -2$
- Each x_i is associated with a weight w_i which determines how important x_i is
  - (For predicting the positive class)

Features in Classification



- Examples of feature x_i
  - $x_i =$ "review contains 'awesome'"; $w_i = +10$
  - $x_j =$ "review contains 'abysmal'"; $w_j = -10$
  - $x_k =$ "review contains 'mediocre'"; $w_k = -2$
- Each x_i is associated with a weight w_i which determines how important x_i is
  - (For predicting the positive class)
- May be
  - manually configured or
  - automatically inferred, as in modern architectures

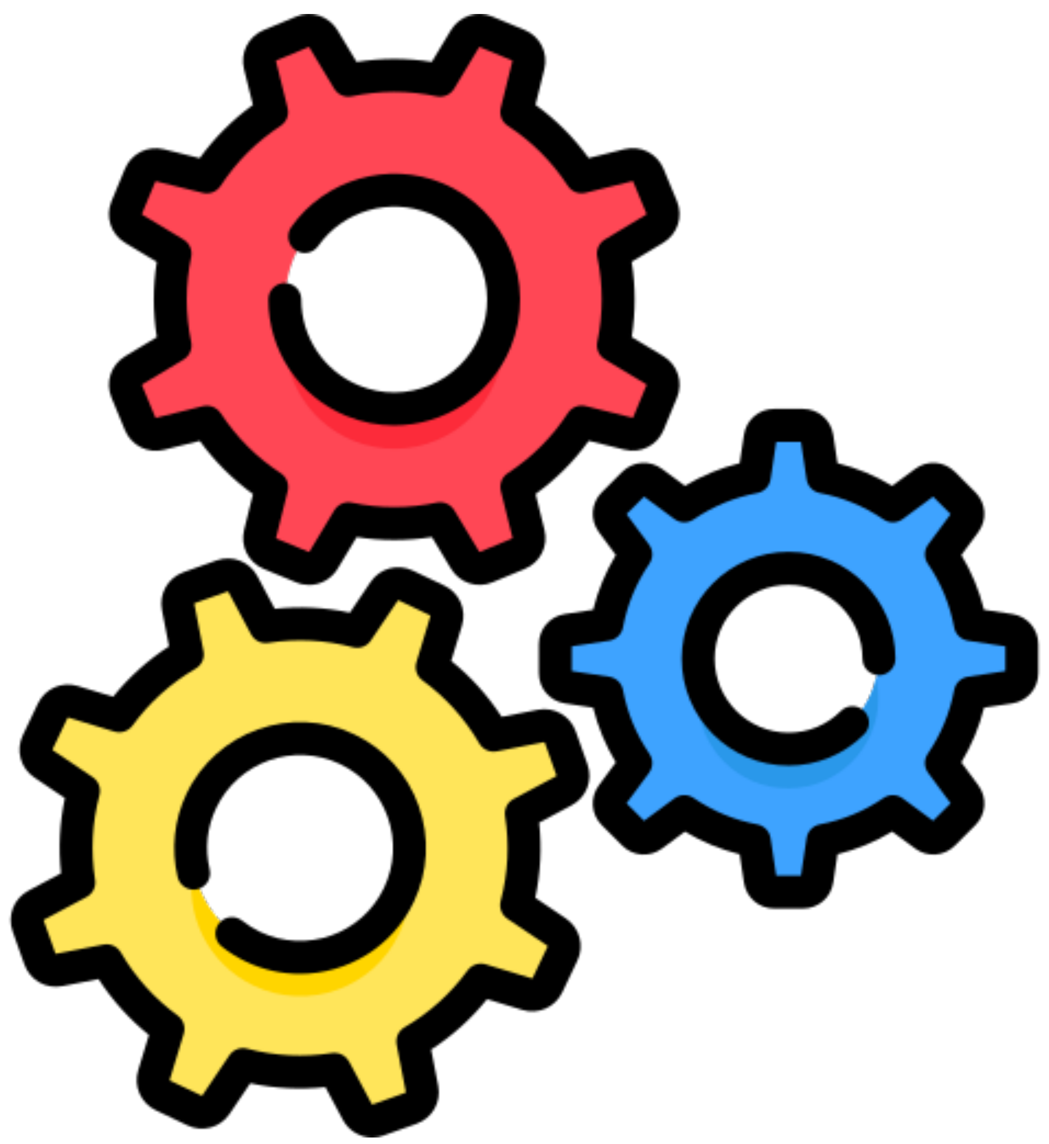
Features in Classification



- Examples of feature x_i
  - $x_i =$"review contains 'awesome'"; $w_i = +10$
  - $x_j =$"review contains 'abysmal'"; $w_j = -10$
  - $x_k =$"review contains 'mediocre'"; $w_k = -2$
- Each x_i is associated with a weight w_i which determines how important x_i is
  - (For predicting the positive class)
- May be
  - manually configured or
  - automatically inferred, as in modern architectures

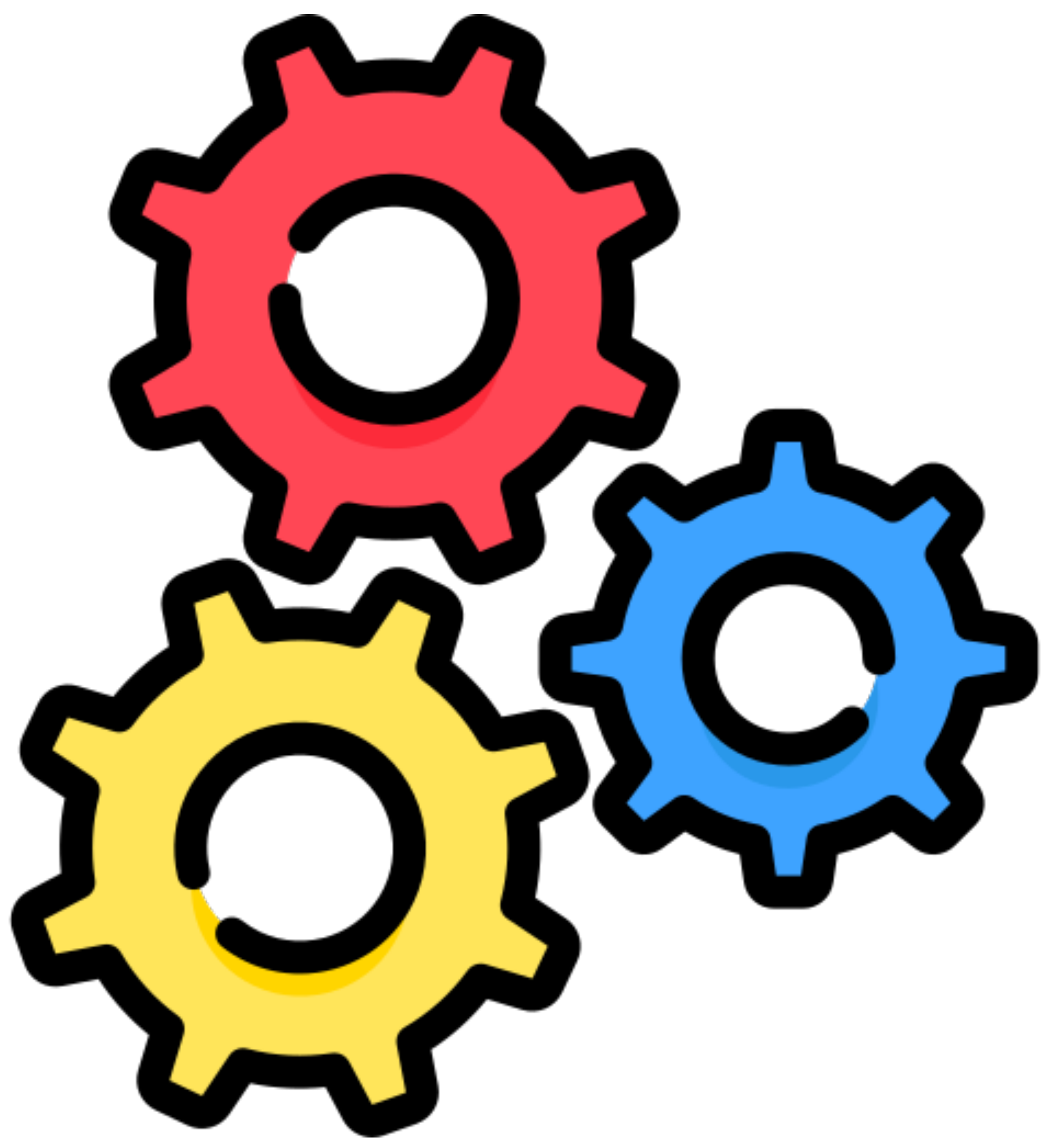
Can you guess the w for x_l = "review contains 'restaurant'"?

Data Pre-processing



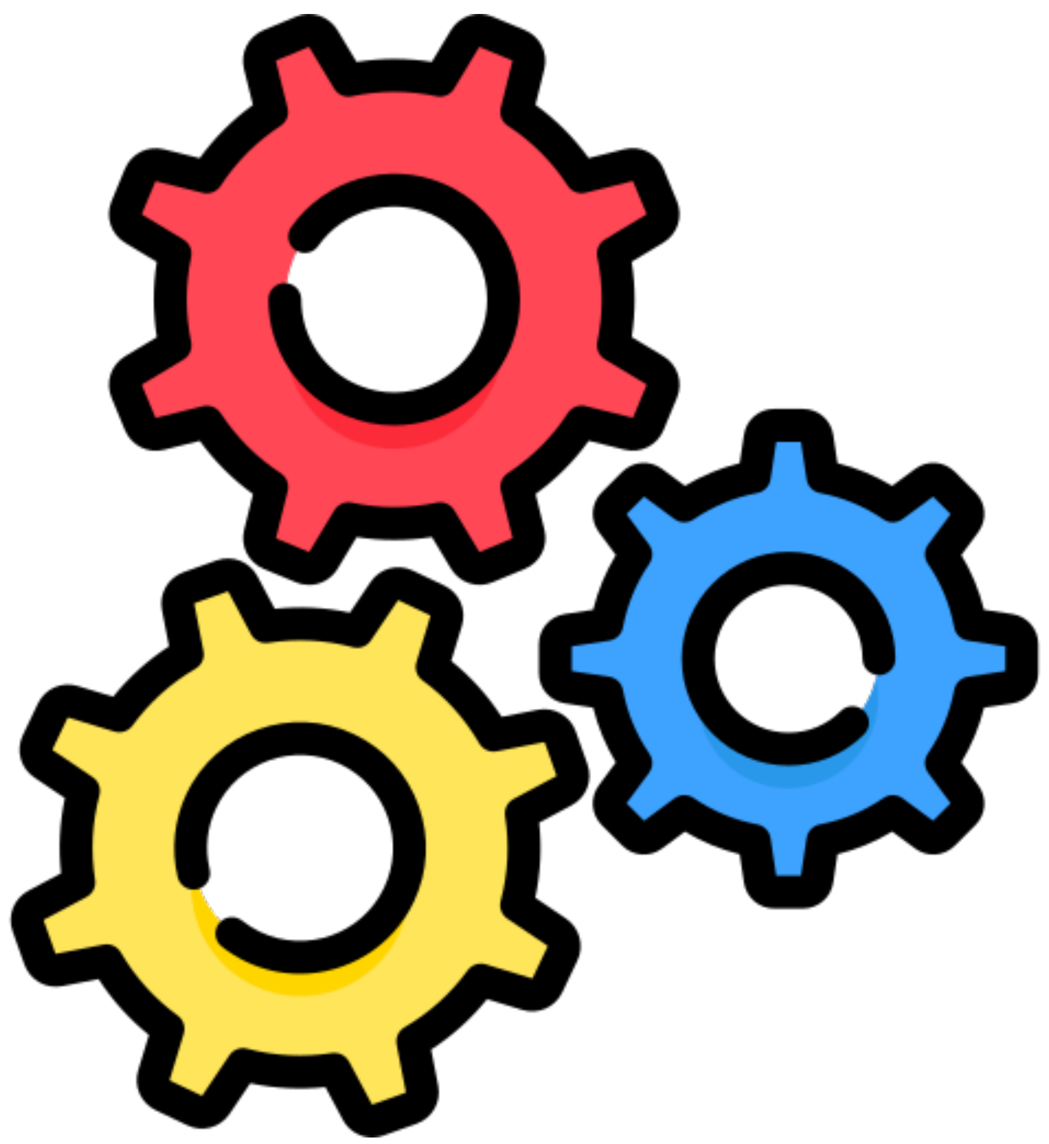
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction



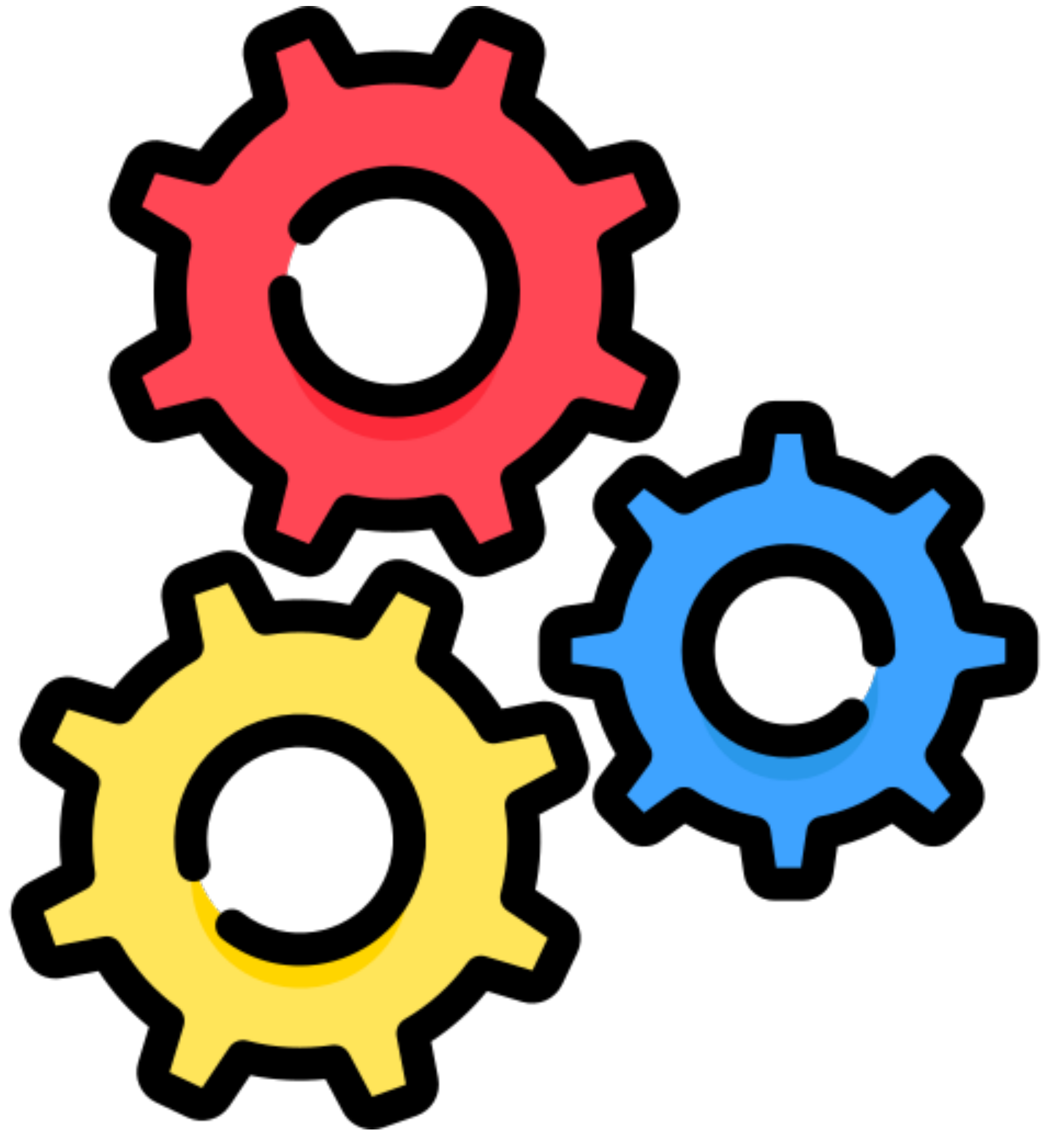
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing



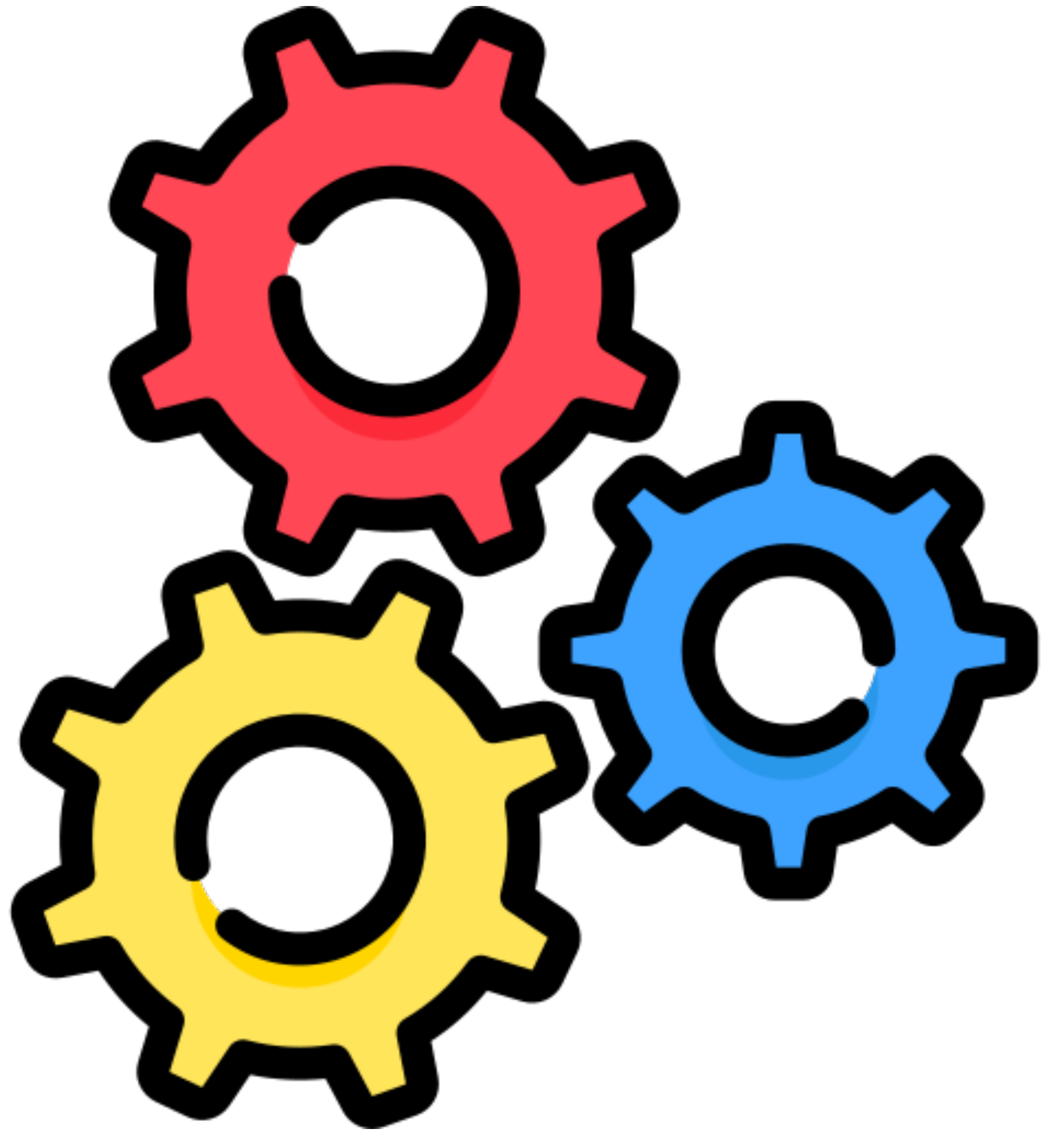
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing
    - Removing extra spaces
    - Removing unhelpful tokens, e.g., external URL links
    - Removing unhelpful characters, e.g., non-alphabetical characters
    - Splitting punctuations: Happy New Year! -> Happy New Year !



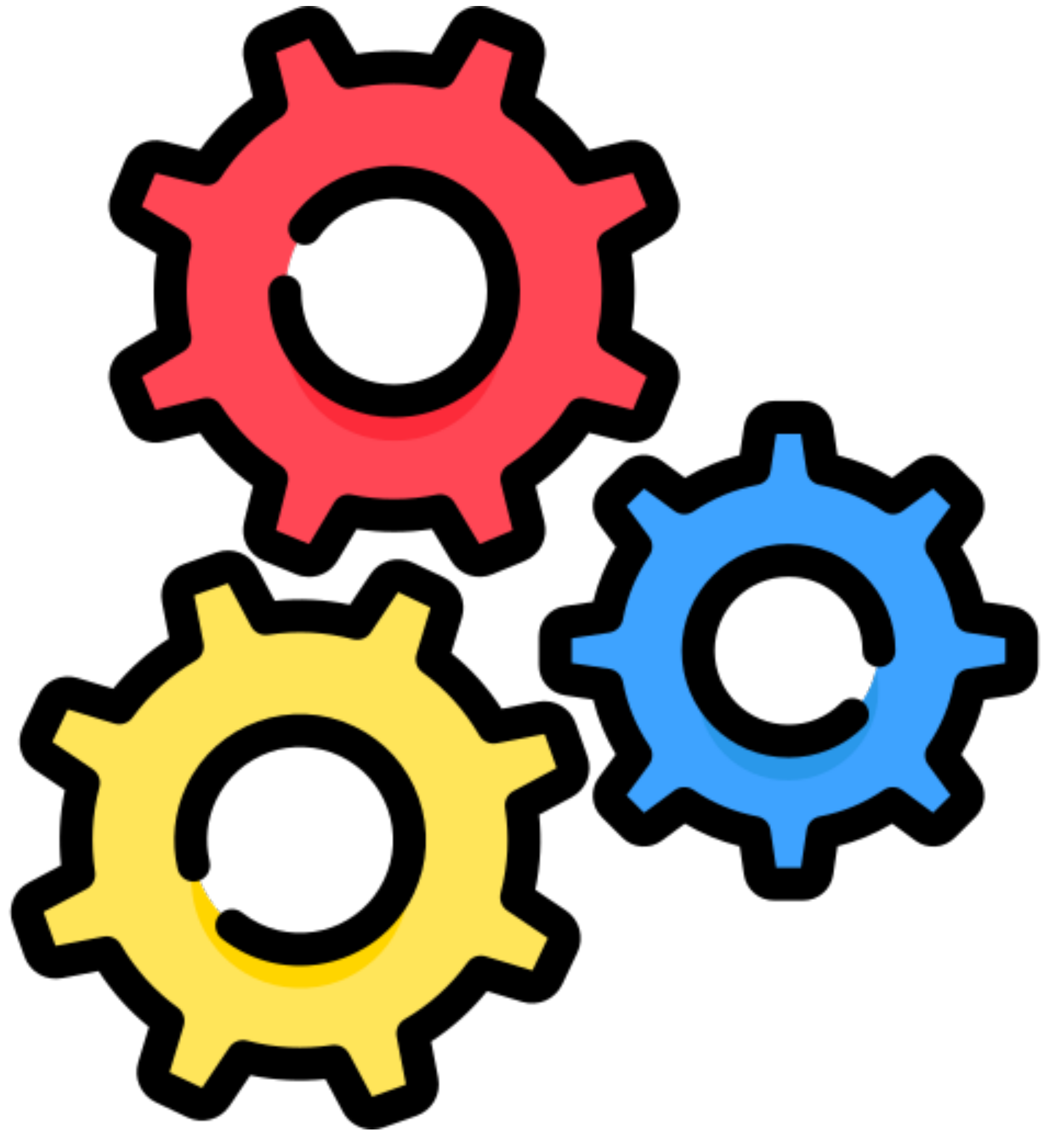
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing
    - Removing extra spaces
    - Removing unhelpful tokens, e.g., external URL links
    - Removing unhelpful characters, e.g., non-alphabetical characters
    - Splitting punctuations: Happy New Year! -> Happy New Year !
  - Optional Steps
    - Contracting and standardizing: won't -> will not



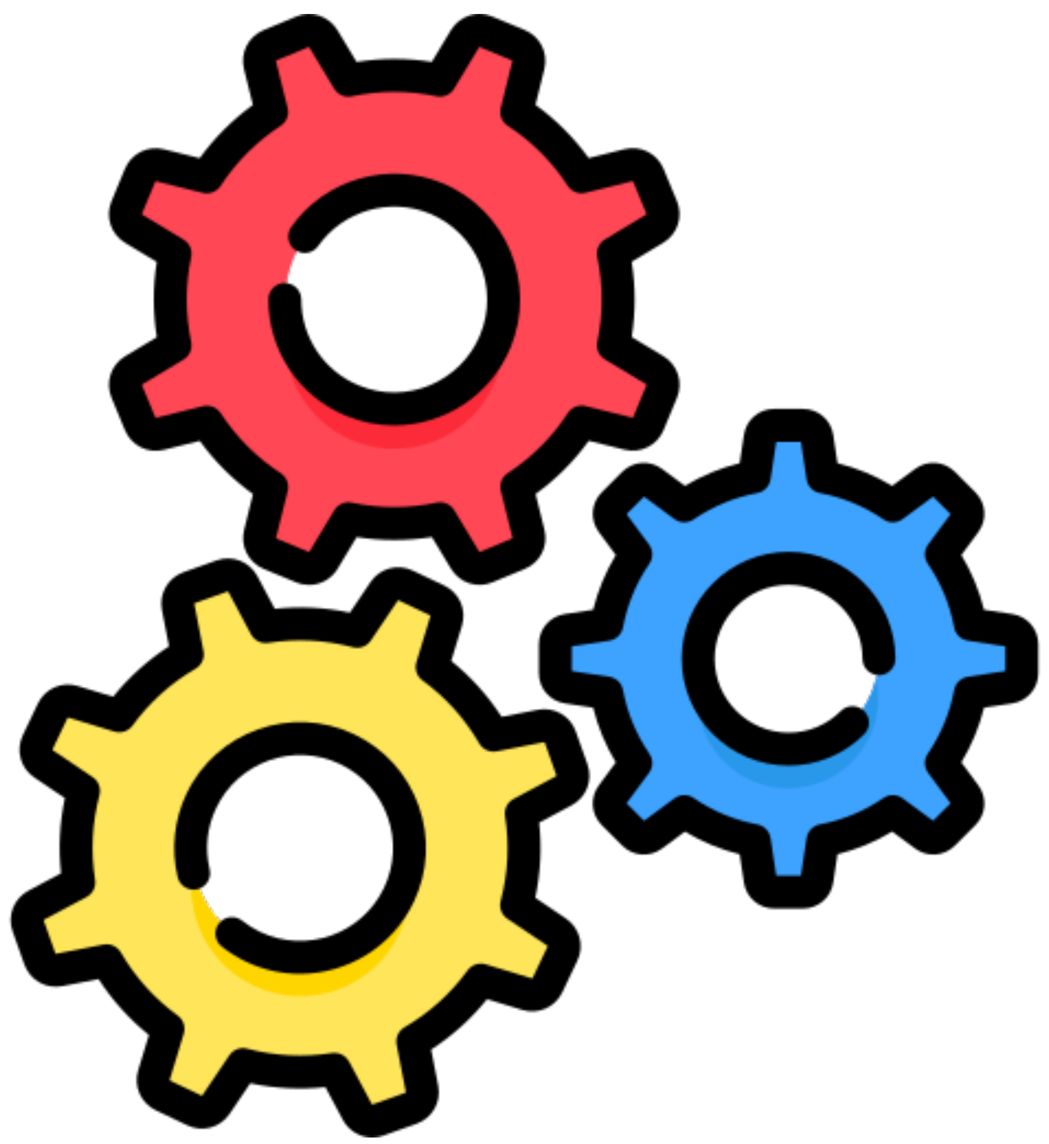
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing
    - Removing extra spaces
    - Removing unhelpful tokens, e.g., external URL links
    - Removing unhelpful characters, e.g., non-alphabetical characters
    - Splitting punctuations: Happy New Year! -> Happy New Year !
  - Optional Steps
    - Contracting and standardizing: won't -> will not
    - Converting capital letters to lowercase: New York -> new york



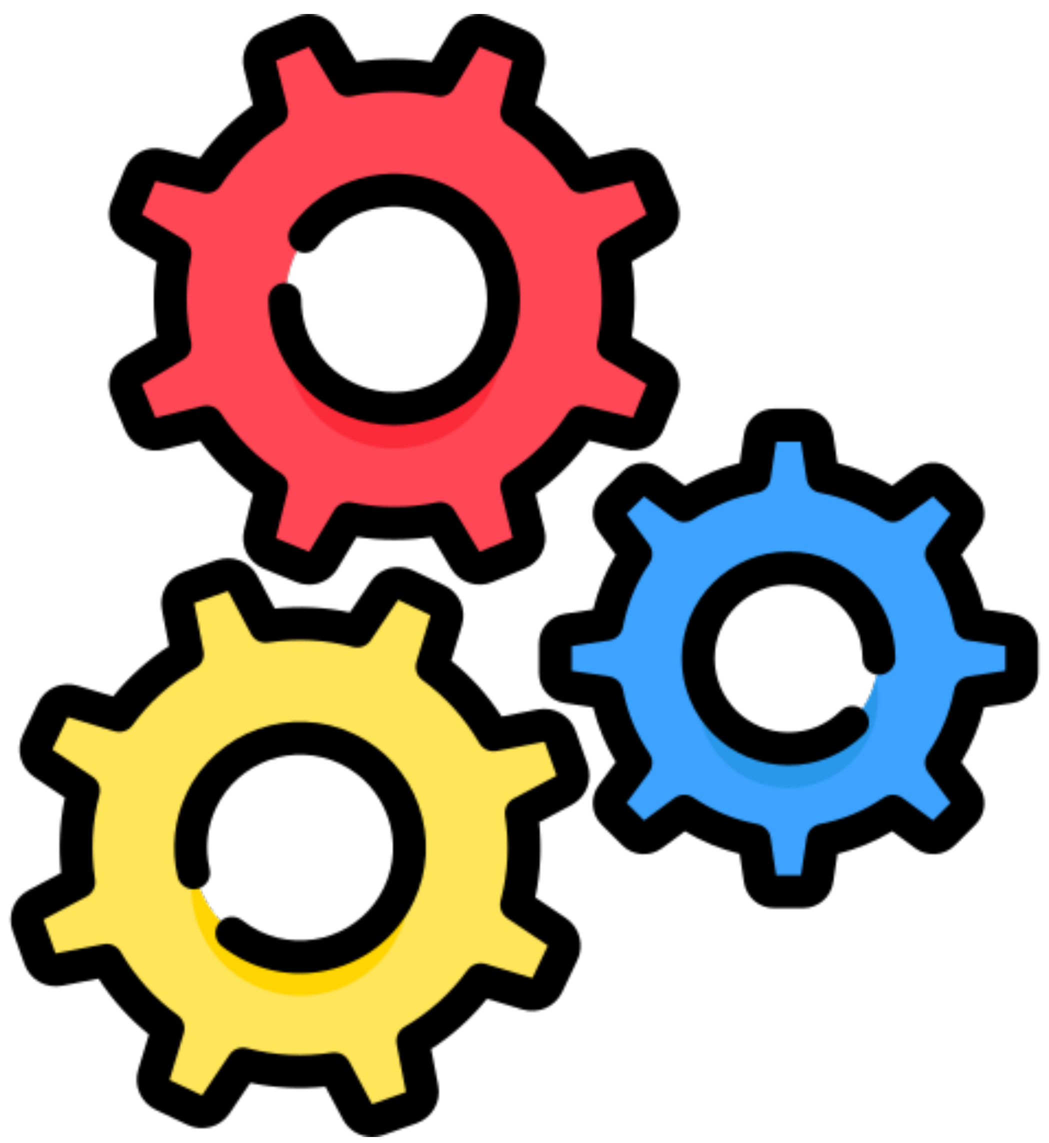
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing
    - Removing extra spaces
    - Removing unhelpful tokens, e.g., external URL links
    - Removing unhelpful characters, e.g., non-alphabetical characters
    - Splitting punctuations: Happy New Year! -> Happy New Year !
  - Optional Steps
    - Contracting and standardizing: won't -> will not
    - Converting capital letters to lowercase: New York -> new york
    - Stemming or Lemmatization: tokenization -> token



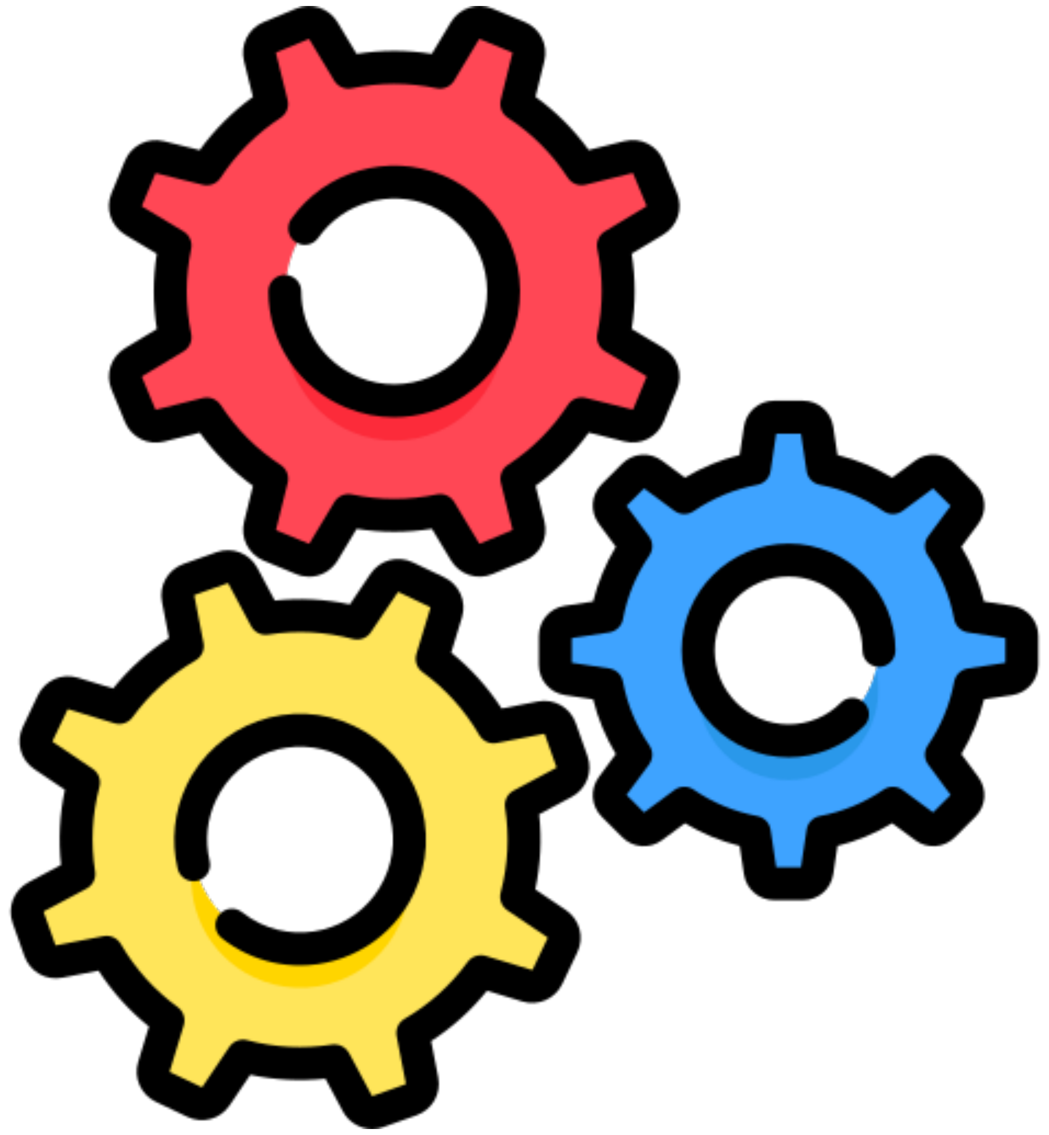
Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing
    - Removing extra spaces
    - Removing unhelpful tokens, e.g., external URL links
    - Removing unhelpful characters, e.g., non-alphabetical characters
    - Splitting punctuations: Happy New Year! -> Happy New Year !
  - Optional Steps
    - Contracting and standardizing: won't -> will not
    - Converting capital letters to lowercase: New York -> new york
    - Stemming or Lemmatization: tokenization -> token
  - In many of the above cases, pre-existing libraries exist for the task (e.g. Porter Stemmer)



Data Pre-processing

- Documents containing raw texts must be preprocessed before feature extraction
  - Tokenization: splitting the text into units for processing
    - Removing extra spaces
    - Removing unhelpful tokens, e.g., external URL links
    - Removing unhelpful characters, e.g., non-alphabetical characters
    - Splitting punctuations: Happy New Year! -> Happy New Year !
  - Optional Steps
    - Contracting and standardizing: won't -> will not
    - Converting capital letters to lowercase: New York -> new york
    - Stemming or Lemmatization: tokenization -> token
  - In many of the above cases, pre-existing libraries exist for the task (e.g. Porter Stemmer)



Still relevant, especially for you to understand what current LLMs can automate!

Feature Extraction

Feature Extraction

- Vocabulary Creation
  - A dictionary of all the words we care about
    - Excluding stop words from dictionary as they are useless for the task at hand
  - Mapping each word to a word id: are -> 2
  - Discarding words not included in the vocabulary

Feature Extraction

- Vocabulary Creation
  - A dictionary of all the words we care about
    - Excluding stop words from dictionary as they are useless for the task at hand
  - Mapping each word to a word id: are -> 2
  - Discarding words not included in the vocabulary

- Feature Representations:
    - Bag of Words (BoW)
    - Term Frequency / Inverse Document Frequency (TF-IDF)
    - Word Embeddings (e.g., word2vec, GloVe)

Feature Extraction

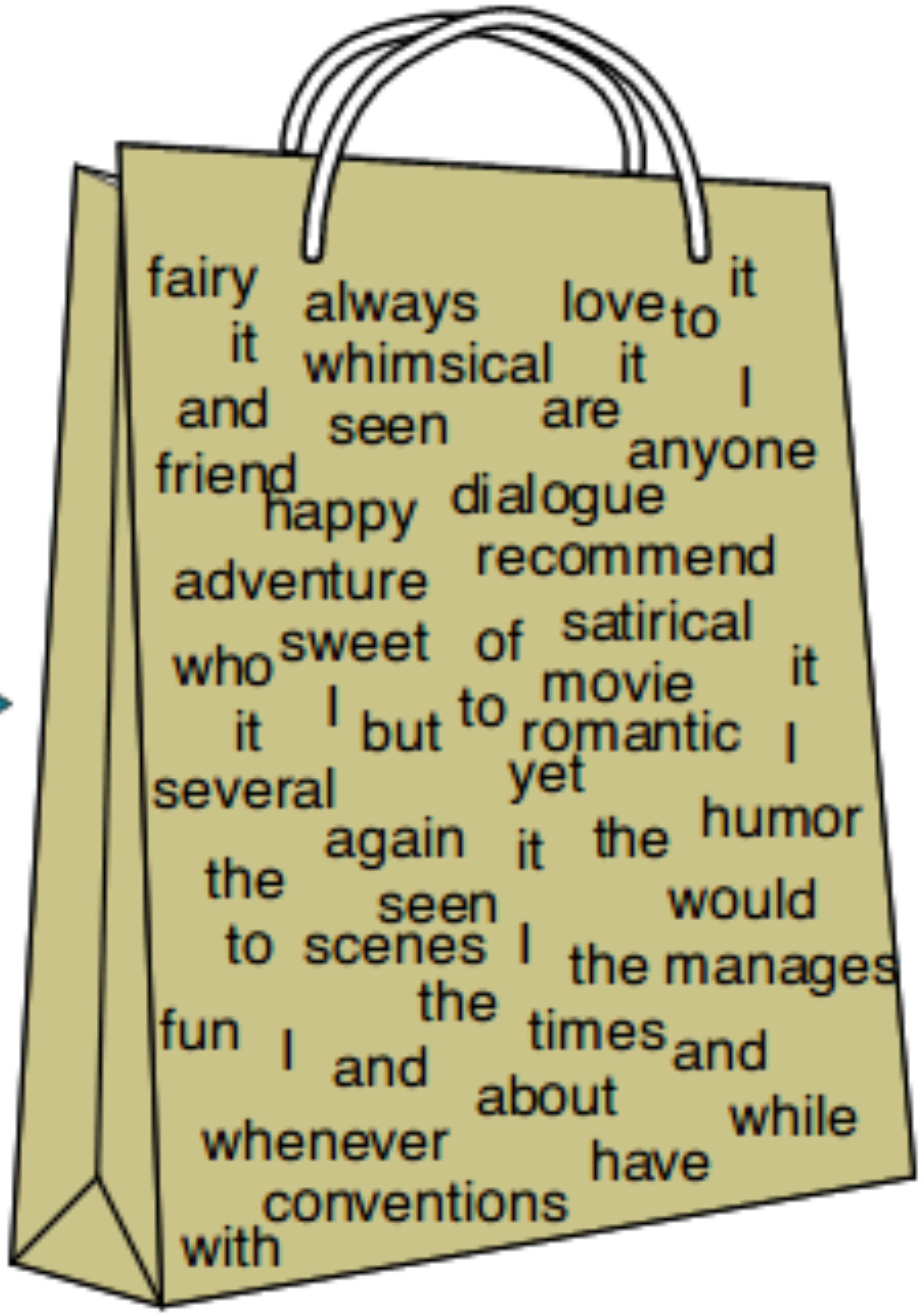
- Vocabulary Creation
  - A dictionary of all the words we care about
    - Excluding stop words from dictionary as they are useless for the task at hand
  - Mapping each word to a word id: are -> 2
  - Discarding words not included in the vocabulary

- Feature Representations:
    - Bag of Words (BoW)
    - Term Frequency / Inverse Document Frequency (TF-IDF)
    - Word Embeddings (e.g., word2vec, GloVe)

What happens when we see OOV words at test time?

Feature Representation: Bag of Words

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
...	...

Bag of Words

Bag of Words

- With a word vocabulary of k words, BoW represents each doc / review $\mathbf{x}$ into a vector of integers

Bag of Words

- With a word vocabulary of k words, BoW represents each doc / review $\mathbf{x}$ into a vector of integers

Vocab = [good, bad, nice, expensive, love]

"I love this shirt because it is nice and warm. The fabric is also nice and the color complements my skin tone."

Bag of Words

- With a word vocabulary of k words, BoW represents each doc / review $\mathbf{x}$ into a vector of integers
- $\mathbf{x} = [x_1, \ldots, x_k], \quad x_i \in 0,1,2,\ldots$
  - $x_i = j$ indicates that word i appears j times in the doc / review $\mathbf{x}$

Vocab = [good, bad, nice, expensive, love]

"I love this shirt because it is nice and warm. The fabric is also nice and the color complements my skin tone."

[0, 0, 2, 0, 1]

Bag of Words

- With a word vocabulary of k words, BoW represents each doc / review $\mathbf{x}$ into a vector of integers
- $\mathbf{x} = [x_1, \ldots, x_k], \quad x_i \in 0,1,2,\ldots$
    - $x_i = j$ indicates that word i appears j times in the doc / review $\mathbf{x}$

Vocab = [good, bad, nice, expensive, love]

"I love this shirt because it is nice and warm. The fabric is also nice and the color complements my skin tone." → **[0, 0, 2, 0, 1]**

Is k the number of types or tokens?

Bag of Words: Pros and Cons

Bag of Words: Pros and Cons

- Limitations:
  - Insensitive to language structure: all contextual information has been discarded
  - Information in word dependencies is overlooked: new york vs new book
  - The resulting vectors are just word counts and are highly sparse
  - Dominated by common words

Bag of Words: Pros and Cons

- Limitations:
  - Insensitive to language structure: all contextual information has been discarded
  - Information in word dependencies is overlooked: new york vs new book
  - The resulting vectors are just word counts and are highly sparse
  - Dominated by common words

Solutions?

Bag of Words: Pros and Cons

- Limitations:
  - Insensitive to language structure: all contextual information has been discarded
  - Information in word dependencies is overlooked: new york vs new book
  - The resulting vectors are just word counts and are highly sparse
  - Dominated by common words

Solutions?

- Pros:
  - Simple!
  - Leads to acceptable performance in quite a few settings

Next Class:
II. Model:
(a) Logistic Regression