Bag-of-words and naive classification

NLP Week 3

Thanks to Dan Jurafsky for some of the slides and inspiration!

Plan for today

1. **Bag-of-words language models**
2. **Text classification and supervised learning**
3. **Naive Bayes Classifiers**
4. ***Group exercises***

What is a language model?

- They are probability models over contextualized language. They can assign probability to words in context and can thus do things like predict the next word in a sentence, or give you the probability of a whole sentence under the model.

- LLM = Large **language model**

This semester

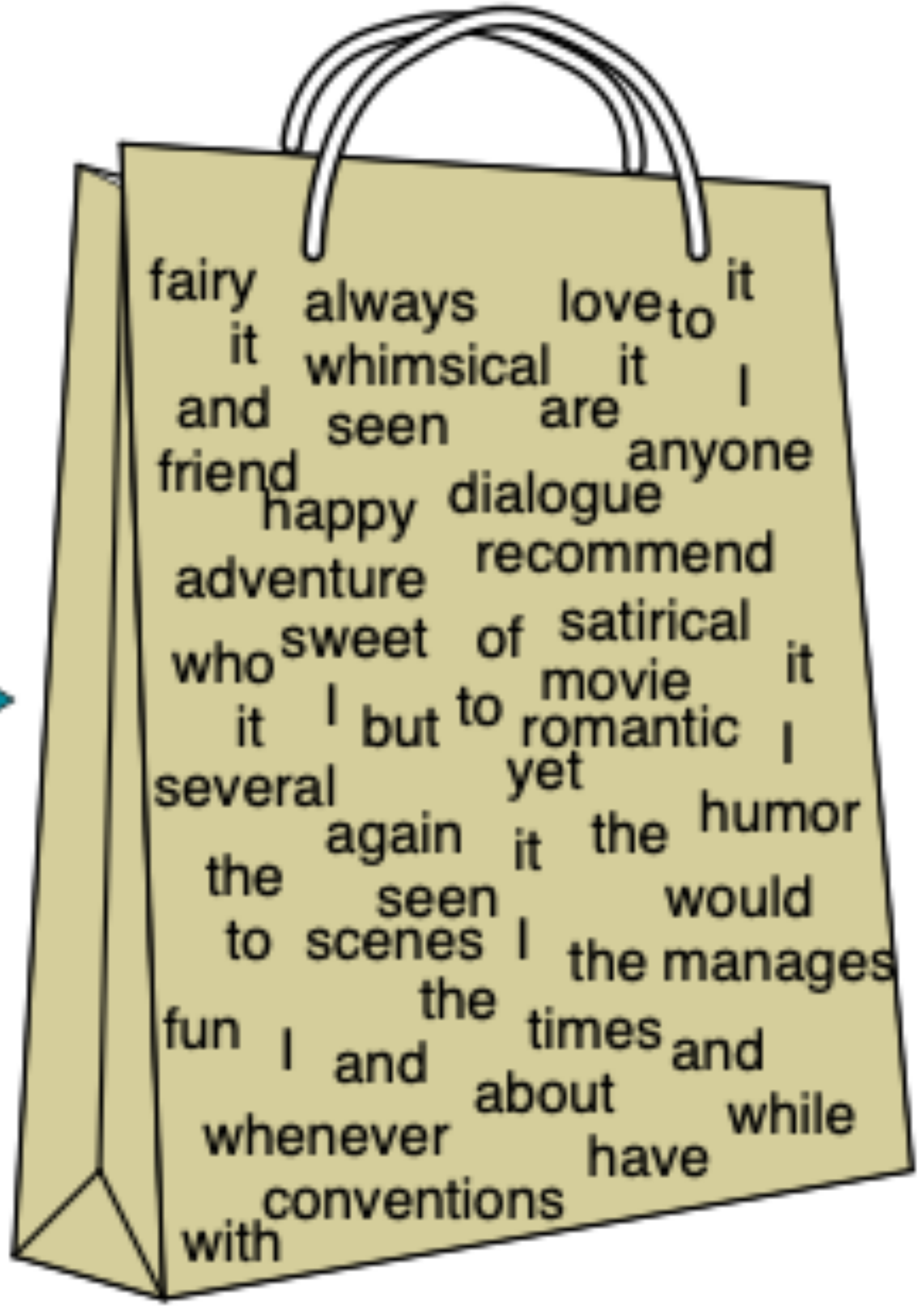
We will build language models adding to each layer of their complexity:

1. **Bag of words models** (basic statistical models of language)
2. **N-gram models** (+ sequential dependencies)
3. **Hidden Markov models** (+ latent categories)
4. **Recurrent neural networks** (+ distributed representations)
5. **LSTM language models** (+ long distance dependencies)
6. **Transformer language models** (+ attention-based dependency learning)

= Today’s language models!

Bag-of-words language model

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!



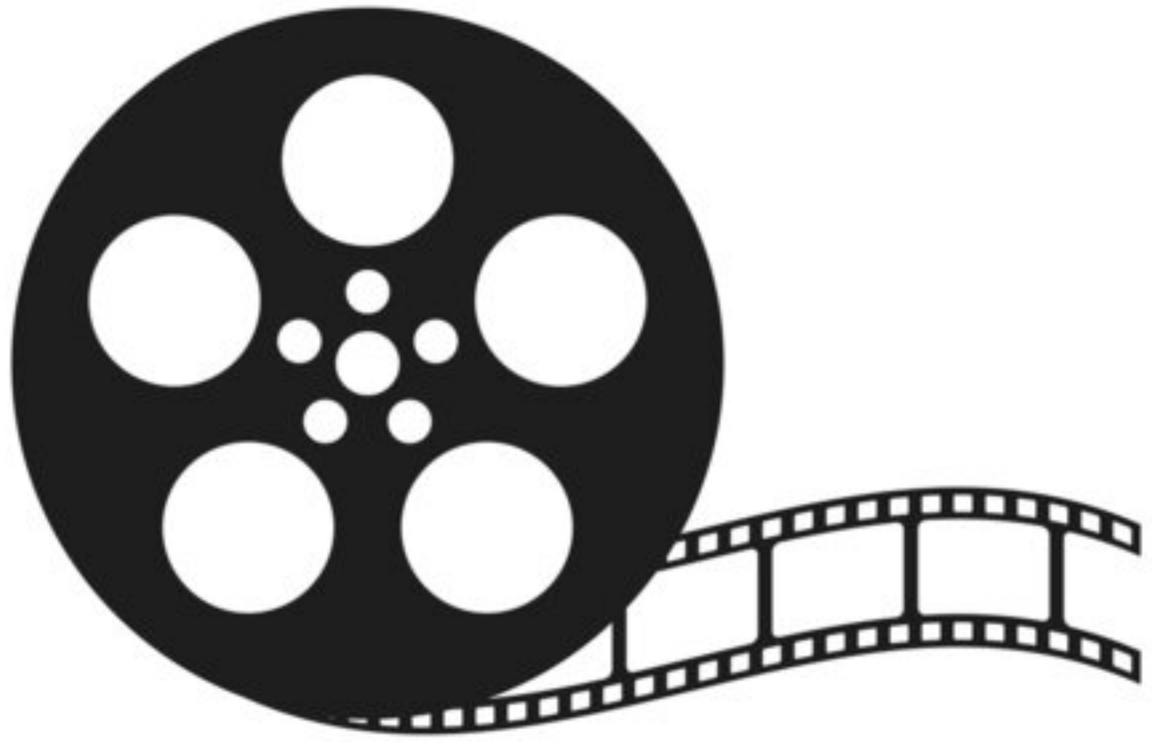
it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Bag-of-words language model

- Bag-of-words model (BoW) also known as a unigram model is a language model which assigns probability to words based on no prior context.

- So the probability of a word is simply its frequency, or normalized count in a document or corpus.

Text classification



Positive or negative movie review?

 unbelievably disappointing

 Full of zany characters and richly applied satire, and some great plot twists

 this is the greatest screwball comedy ever filmed

 It was pathetic. The worst part about it was the boxing scenes.

Text classification

- **Other examples:**
  - Review classification
  - Spam detection
  - Sentiment analysis
  - Author detection
  - Topic/subject assignment

… and the list goes on!

Text classification

Text classification

Text classification is…

Text classification

Text classification is…

The task of assigning a class label to a document or text.

Text classification

Text classification is…

The task of assigning a class label to a document or text.

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: a predicted class $c \in C$

Text classification

Text classification is...

The task of assigning a class label to a document or text.

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: a predicted class $c \in C$

Text classification

Text classification is…

The task of assigning a class label to a document or text.

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: a predicted class $c \in C$

Text classification

Text classification is…

The task of assigning a class label to a document or text.

Input:

- a document d <- "This was the greatest comedy of the year"
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: a predicted class $c \in C$

Text classification

Text classification is…

The task of assigning a class label to a document or text.

Input:

- a document d <- “This was the greatest comedy of the year”
- a fixed set of classes $C = \{c_1, c_2,\ldots, c_J\}$ <- C = {Positive, Neutral, Negative}

Output: a predicted class $c \in C$

Text classification

What should our classification algorithm look like?

Input

↓

Classifier

↓

Output

Text classification

What should our classification algorithm look like?

“This was the greatest comedy of the year”

Input

↓

Classifier

↓

Output

Text classification

What should our classification algorithm look like?

Input

↓

Classifier

↓

Output

“This was the greatest comedy of the year”

C = {Positive, Negative}

Text classification

What should our classification algorithm look like?

Input

"This was the greatest comedy of the year"

C = {Positive, Negative}

↓

Classifier

↓

Output

c = Positive

Text classification

What should our classification algorithm look like?

Input

“This was the greatest comedy of the year”

C = {Positive, Negative}

↓

Classifier

Heuristic/rule-based

↓

Output

c = Positive

Text classification

What should our classification algorithm look like?

Input	**"This was the greatest comedy of the year"** **C = {Positive, Negative}**	
↓		
Classifier	**Heuristic/rule-based**	
↓		
Output	**c = Positive**	

```
If "greatest" in d:
  Return positive
Else if "worst" in d:
  Return negative
```

Text classification

What should our classification algorithm look like?

Input

But … "This was the greatest flop of the year"

C = {Positive, Negative}

↓

Classifier

Heuristic/rule-based

```
If "greatest" in d:
    Return positive
Else if "worst" in d:
    Return negative
```

↓

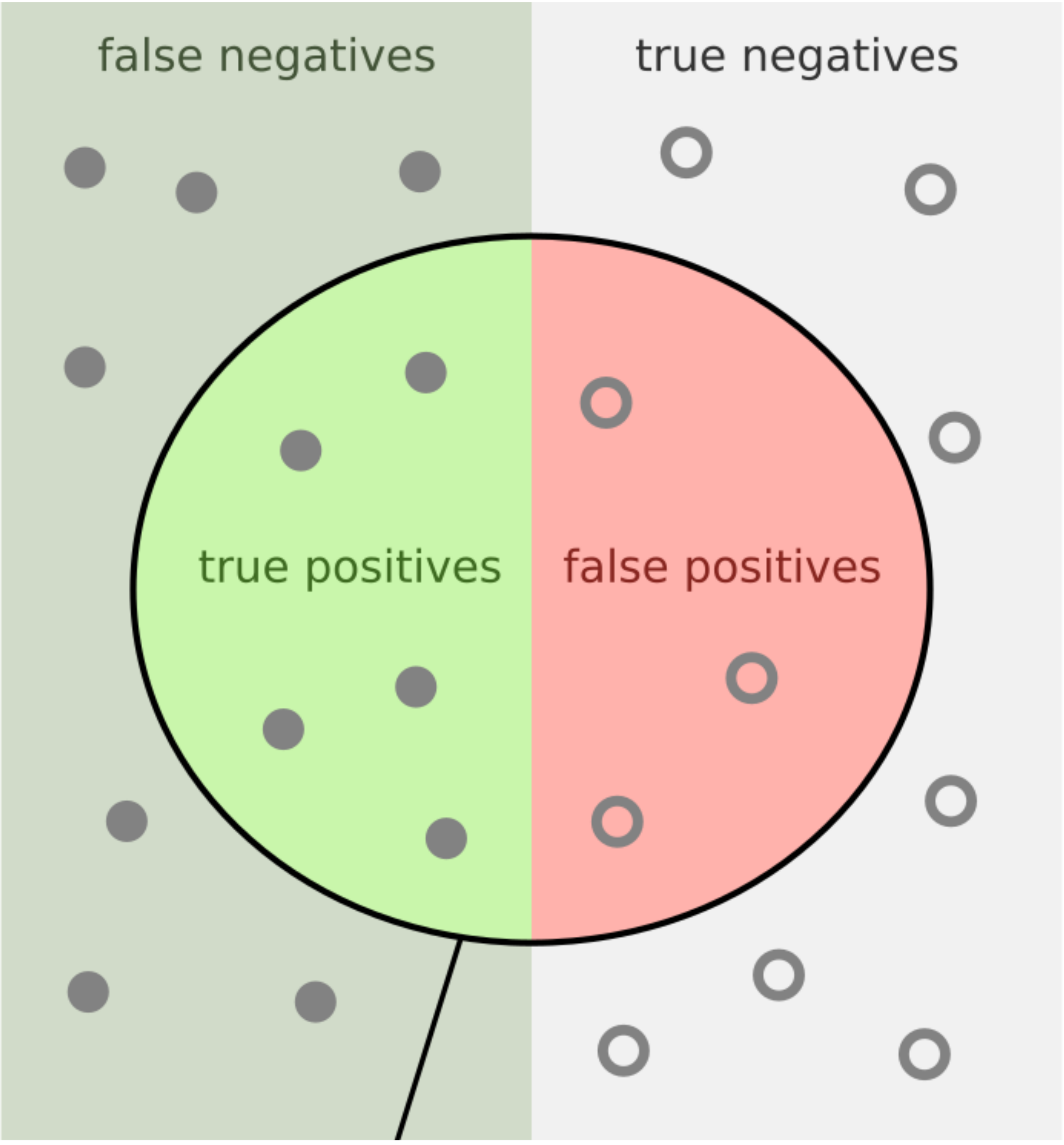
Output

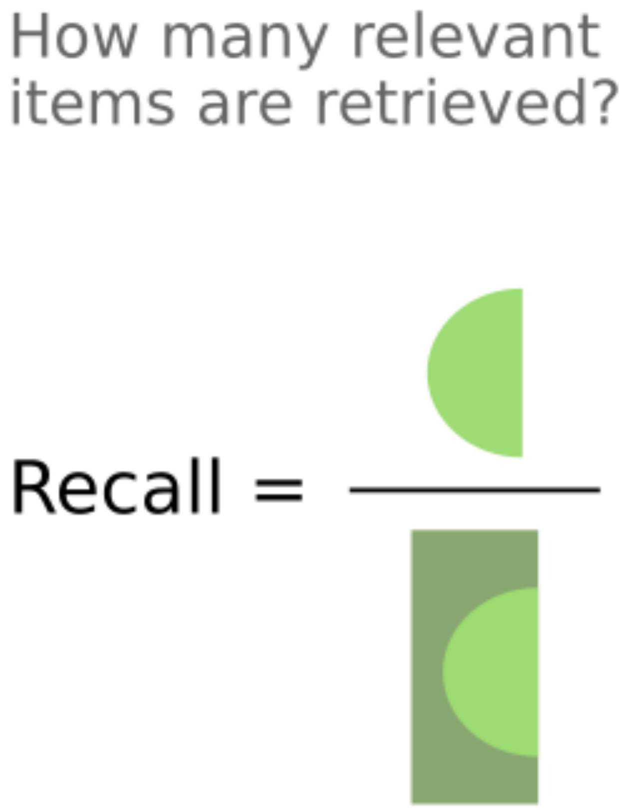
c = Positive

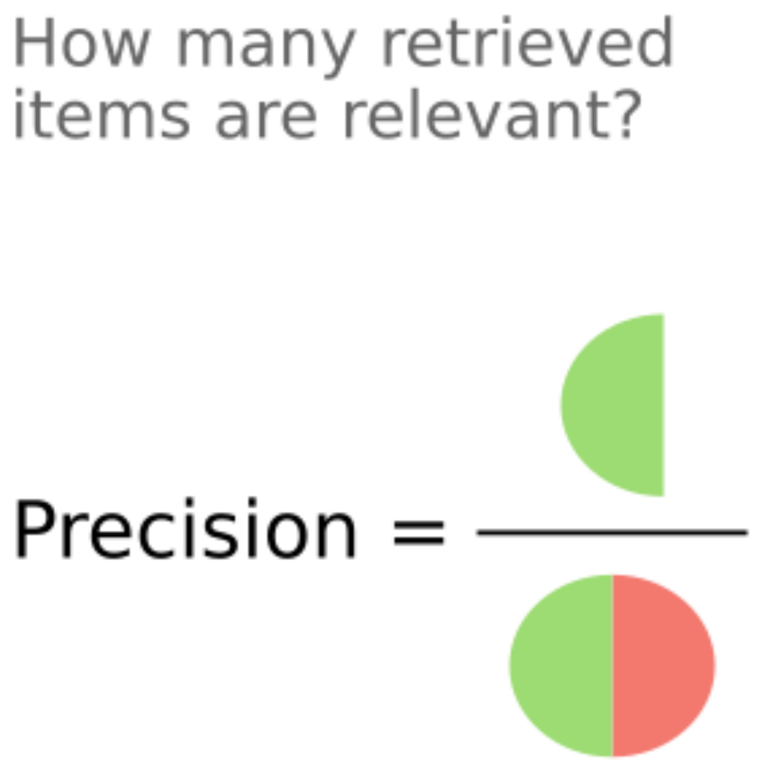
Precision and Recall

- Heuristic classifiers tend to have high **precision** but very low **recall.**

- **Classifier accuracy is measured with precision and recall:**





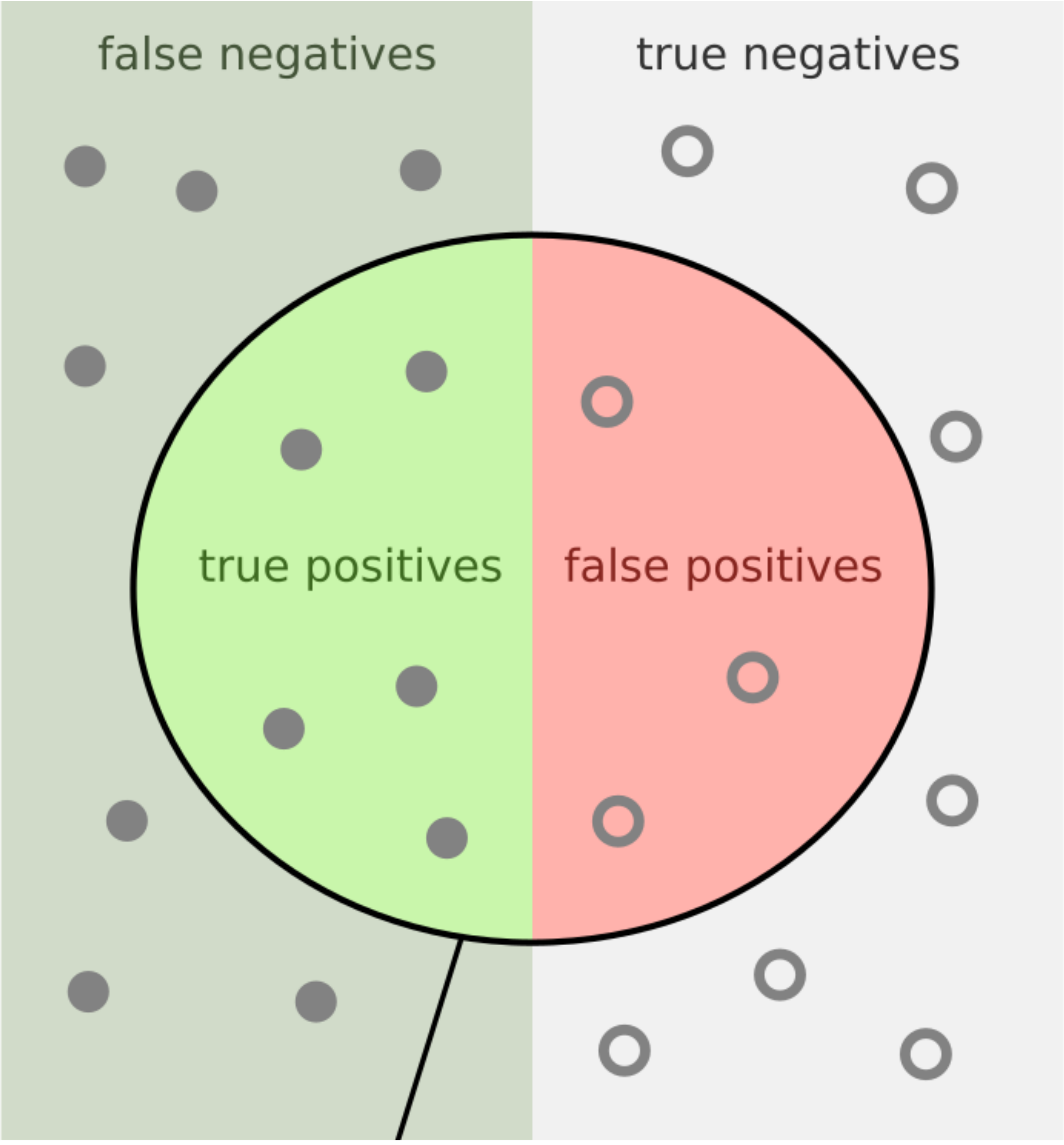


Recall = TP / (**FN** +TP)   Precision = TP / (**FP** +TP)

Precision and Recall

F-score is the harmonic mean of the two:

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$



How many relevant items are retrieved?

Recall = 

How many retrieved items are relevant?

Precision = 

Recall = TP / (**FN** +TP)

Precision = TP / (**FP** +TP)

Supervised learning

An alternative to heuristic classifiers is to use machine learning to ***learn an algorithm rather than define an algorithm*** that will correctly classify texts.

We do this using ***supervised learning***. Supervised learning using a training data which is correctly ***labelled*** with the desired classes. We will then try to learn a set of shared features across documents for each class.

Supervised learning

An alternative to heuristic classifiers is to use machine learning to ***learn an algorithm rather than define an algorithm*** that will correctly classify texts.

We do this using ***supervised learning***. Supervised learning using a training data which is correctly *labelled* with the desired classes. We will then try to learn a set of shared features across documents for each class.

Input:

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

Output:

- a learned classifier $\gamma: d \rightarrow c$

Supervised learning

There are many types of learned classifiers:

- Naïve Bayes (this lecture)
- Logistic regression
- Neural networks
- …
- Even finetuned LLMs or prompted LLMs

All require labelled training data to do classification

Naive Bayes Classifier

Naive Bayes Intuition

Simple ("naive") classification method based on Bayes rule

Relies on very simple representation of document

- **Bag of words**

Bag-of-words representation

- We can represent a document using a bag-of-words representations

$\gamma($

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

$) = c$

Naive Bayes Classifier

Bayes' Rule Applied to Documents and Classes

For a document *d* and a class *c*

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

Naive Bayes Classifier

Naive Bayes Classifier

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

MAP is "maximum a posteriori" = most likely class

Naive Bayes Classifier

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

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

Naive Bayes Classifier

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

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Naive Bayes Classifier

"Likelihood" "Prior"

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

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Bag-of-words representation

- We can represent a document using a bag-of-words representations

$\boldsymbol{d}$ =

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

- Each feature can then be the normalized count of some word in the document!

Naive Bayes Classifier

Multinomial Naive Bayes Independence Assumptions

$$P(x_1, x_2, \ldots, x_n \mid c)$$

Bag of Words assumption: Assume position doesn't matter

Conditional Independence: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Naive Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Naive Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \, P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \, P(c_j) \prod_{x \in X} P(x \mid c)$$

In practice: log probabilities

Problems with multiplying lots of probs

There's a problem with this:

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

Multiplying lots of probabilities can result in floating-point underflow!

.0006 * .0007 * .0009 * .01 * .5 * .000008….

Idea: Use logs, because log(*ab*) = log(*a*) + log(*b*)

We'll sum logs of probabilities instead of multiplying probabilities!

In practice: log probabilities

We actually do everything in log space

Instead of this: $$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

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

Notes:

1) Taking log doesn't change the ranking of classes!

   The class with highest probability also has highest log probability!

2) It's a linear model:

   Just a max of a sum of weights: a **linear** function of the inputs

   So naive bayes is a **linear classifier**

In practice: Laplace smoothing

First attempt: maximum likelihood estimates

- simply use the frequencies in the data

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

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

In practice: Laplace smoothing

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

First attempt: maximum likelihood estimates

- simply use the frequencies in the data

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

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

In practice: Laplace smoothing

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

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

Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

In practice: Laplace smoothing

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c)}{\sum_{w \in V} \left(count(w, c) \right)}$$

In practice: Laplace smoothing

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

In practice: Laplace smoothing

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

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

In practice: Unknown words

What about unknown words

- that appear in our test data
- but not in our training data or vocabulary?

We **ignore** them

- Remove them from the test document!
- Pretend they weren't there!
- Don't include any probability for them at all!

Why don't we build an unknown word model?

- It doesn't help: knowing which class has more unknown words is not generally helpful!

[15 minute break]

Working with BoWs models!

Team up!

Open exercises/week 3 in your course folder and start writing/running code!