

NLP Crash Course I

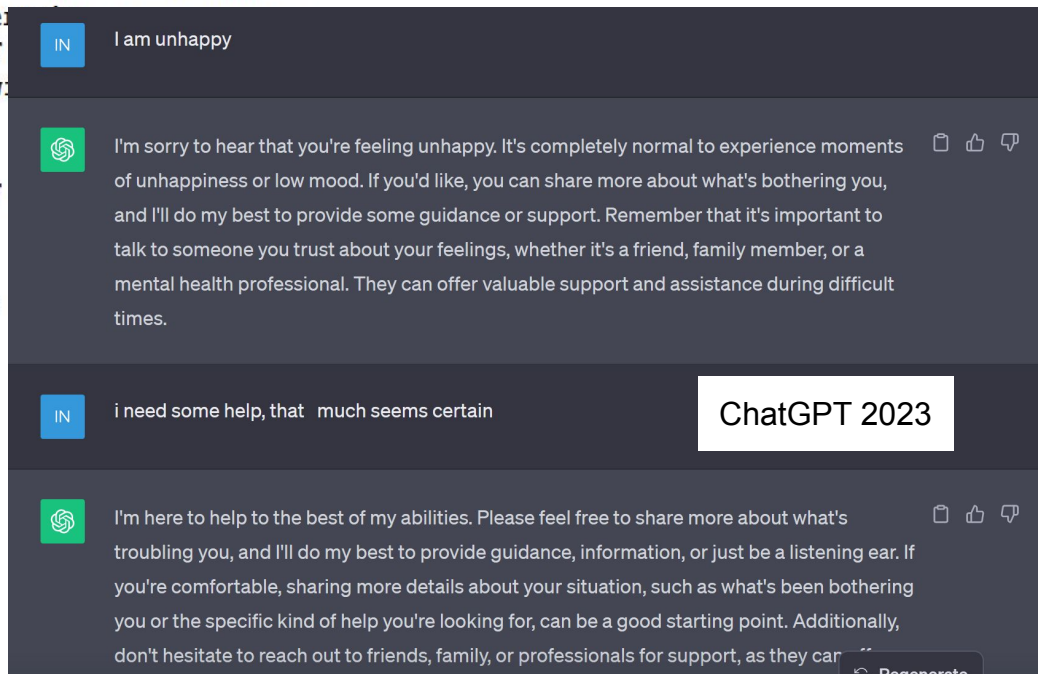
Indira Sen

Agenda

- About Natural Language Processing, it's Origins and Neighbors
- Language Modeling
- Text classification
- Vector Semantics and Embeddings

What is NLP?

User: I am unhappy.
ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User: I need some help, that much seems certain
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT
User: Perhaps I could learn to get along with
ELIZA: TELL ME MORE ABOUT YOUR FAMILY
User: My mother takes care of me.
ELIZA: WHO ELSE IN YOUR FAMILY TAKES CARE OF
User: My father.
ELIZA: YOUR FATHER
User: You are like my father in some ways.



What is NLP?

“ Natural language processing is the set of methods for making human language accessible to computers. ”

- Jacob Eisenstein, Natural Language Processing, First Edition

Neighbors of NLP: Related Area

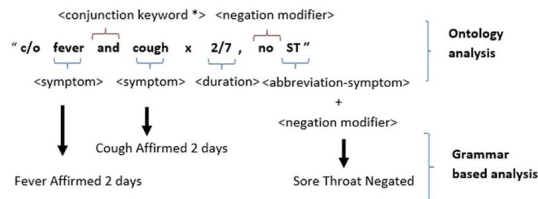
- Computational Linguistics
- Social sciences (sociolinguistics, ethics)
- Computer science
- Machine learning
- ...

What is NLP?

User: I am unhappy.
ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User: I need some help, that much seems certain.
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User: Perhaps I could learn to get along with my mother.
ELIZA: TELL ME MORE ABOUT YOUR FAMILY
User: My mother takes care of me.
ELIZA: WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
User: My father.
ELIZA: YOUR FATHER
User: You are like my father in some ways.

Weizenbaum (1966)

Timeline of NLP



1960s: rule-based approaches

1990-2000s:
Machine Learning
approaches

2020s:
General-purpose
models, ?

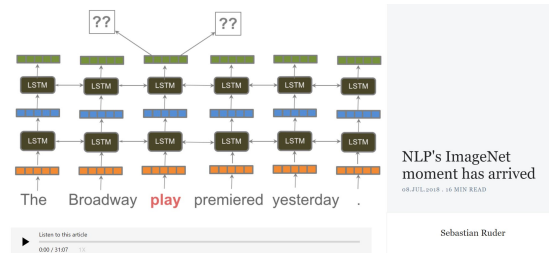
1950s: early
beginnings, Turing

1970-80s: statistical
NLP

2010s: DL as de
facto

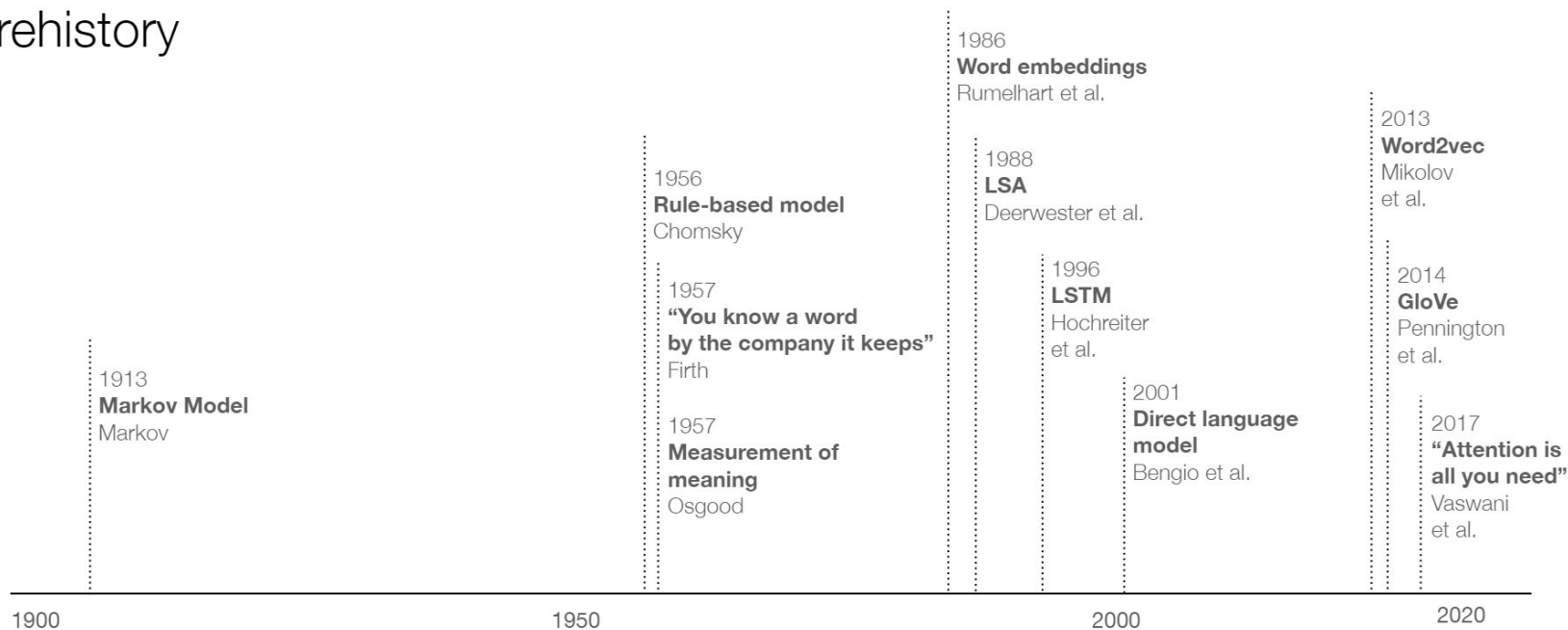
TURING TEST EXTRA CREDIT:
CONVINCE THE EXAMINER
THAT HE'S A COMPUTER.

YOU KNOW, YOU MAKE
SOME REALLY GOOD POINTS.
I'M ... NOT EVEN SURE
WHO I AM ANYMORE.



Language models

Prehistory



Agenda

- About Natural Language Processing, it's Origins and Neighbors
- Language Modeling
- Text classification
- Vector Semantics and Embeddings
- Language Generation

Language Modeling

Q. What does it mean to "model something"?

- To mimic something
- To create a smaller/less complex version of something and study it
- To predict the future behavior of that something

“A good model would simulate the behavior of the real world: it would "understand" which events are in better agreement with the world, i.e., which of them are more likely.”

- Lena Voita, NLP Course for you

Language Modeling

“ Models that assign probabilities to sequences of words are called language models or LMs ”

- Jurafsky and Martin, Speech and Language Processing, Third Edition

Language Modeling

- Helpful for several use cases
 - Machine translation
 - Grammar and spelling correction
 - Speech recognition
 - Search

Language Modeling: Probability + Language

$P(\text{I saw a cat on } \dots) =$

$P(\text{I}) \cdot P(\text{saw}|\text{I}) \cdot P(\text{a}|\text{I saw}) \cdot P(\text{cat}|\text{I saw a}) \cdot P(\text{on}|\text{I saw a cat}) \cdot \dots$

Probability of **I** saw a cat on



Language Modeling: Probability + Language

$P(\text{I saw a cat on } \dots) =$

$$P(\text{I}) \cdot P(\text{saw}|\text{I}) \cdot P(\text{a}|\text{I saw}) \cdot P(\text{cat}|\text{I saw a}) \cdot P(\text{on}|\text{I saw a cat}) \cdot \dots$$

Probability of I saw a cat on



Formally, let y_1, y_2, \dots, y_n be tokens in a sentence, and $P(y_1, y_2, \dots, y_n)$ the probability to see all these tokens (in this order). Using the product rule of probability (aka the chain rule), we get

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^n P(y_t|y_{<t}).$$

N-Gram Language Model

How do we actually get the probabilities in a language model?

Two broad approaches:

- N-gram Language Models
- Neural Models

N-grams?

Probabilistic Language Modeling

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

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$$

- Related task: probability of an upcoming word:

$$P(w_5 | w_1, w_2, w_3, w_4)$$

- A model that computes either of these:

$P(W)$ or $P(w_n | w_1, w_2 \dots w_{n-1})$ is called a **language model**.

How to compute $P(W)$

- How to compute this joint probability:
 - $P(\text{its, water, is, so, transparent, that})$
- Intuition: let's rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

For two **events** A and B , the chain rule states that

$$\mathbb{P}(A \cap B) = \mathbb{P}(B \mid A)\mathbb{P}(A),$$

where $\mathbb{P}(B \mid A)$ denotes the **conditional probability** of B given A .

- More variables:

$$P(A,B,C,D) = P(A)P(B \mid A)P(C \mid A,B)P(D \mid A,B,C)$$

- The Chain Rule in General

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \dots P(x_n \mid x_1, \dots, x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

$P(\text{"its water is so transparent"}) =$

$P(\text{its}) \times P(\text{water} \mid \text{its}) \times P(\text{is} \mid \text{its water})$

$\times P(\text{so} \mid \text{its water is}) \times P(\text{transparent} \mid \text{its water is so})$

How to estimate these probabilities

- Could we just count and divide?

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

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption



Andrei Markov

- Simplifying assumption:

$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$

- Or maybe

$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$

Markov Assumption

$$P(w_1 w_2 \dots w_n) \approx \prod P(w_i \mid w_{i-k} \dots w_{i-1})$$

- In other words, we approximate each component in the product

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

Simplest case: Unigram model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \dots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
 - because language has **long-distance dependencies**:

“The computer which I had just put into the machine room on the fifth floor crashed.”
- But we can often get away with N-gram models

Estimating bigram probabilities

- The Maximum Likelihood Estimate

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

An example

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

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

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67$$

$$P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33$$

$$P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

More examples: Berkeley Restaurant Project sentences

- 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

Raw bigram counts

- Out of 9222 sentences

| | 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 | - | - | - | - | - | - |

- Dan Jurafsky, <https://web.stanford.edu/~jurafsky/NLPCourseSlides.html>

Raw bigram probabilities

- Normalize by unigrams:

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

- Result:

| | 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 |

Bigram estimates of sentence probabilities

$$\begin{aligned} P(<s> \text{ I want english food } </s>) = \\ & P(\text{I} | <s>) \\ & \times P(\text{want} | \text{I}) \\ & \times P(\text{english} | \text{want}) \\ & \times P(\text{food} | \text{english}) \\ & \times P(</s> | \text{food}) \\ & = .000031 \end{aligned}$$

Practical Issues

- We do everything in log space
 - Avoid underflow
 - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Google N-Gram Release, August 2006

AUG

3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word **n-gram models** for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensable 40
- serve as the individual 234

<http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html>

Google Book N-grams

- <http://ngrams.googlelabs.com/>

Evaluating LMs (and other NLP components)

- Extrinsic evaluation
- Intrinsic evaluation

Short Detour: Machine Learning

What is Machine Learning?

"Field of study that gives computers the ability to learn without being explicitly programmed" Arthur Samuel(1959)

"A computer program is said to **learn** from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E ." Tom Mitchell (1998)



Short Detour: Machine Learning

"A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E ."

Your Mail programm "observes" which mails you do or do not classify as spam, and uses these observations to learn how to better filter spam messages. What is the task in this setting?

Classifying mails into the categories "spam" or "not spam"

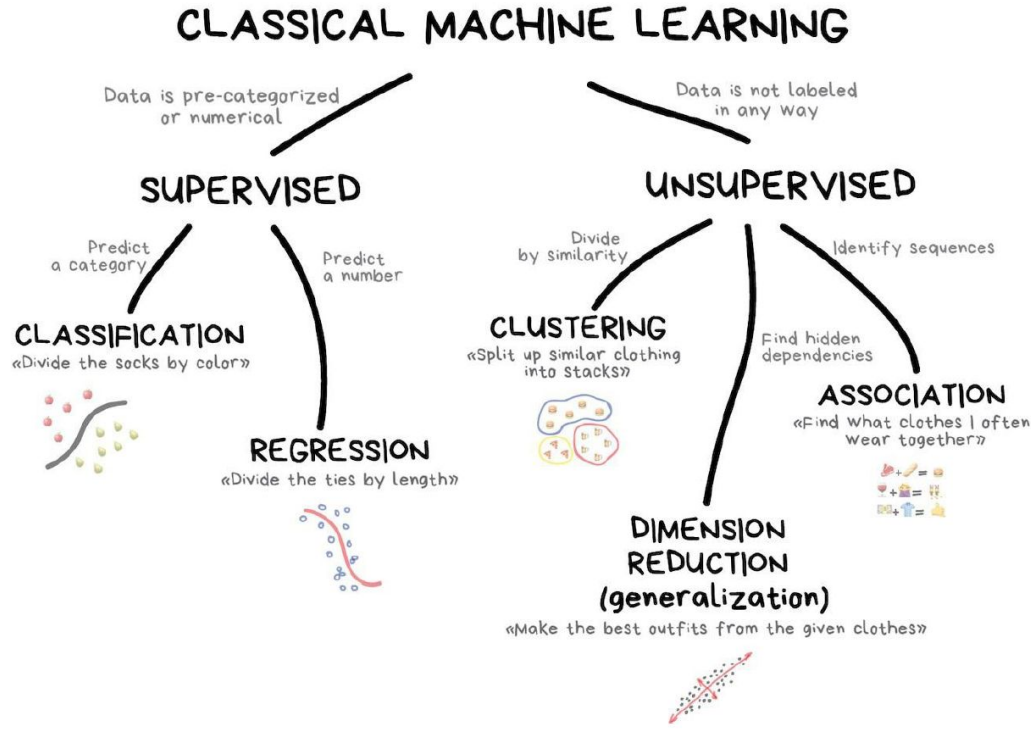
Short Detour: Machine Learning

"A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E ."

Your Mail programm "observes" which mails you do or do not classify as spam, and uses these observations to learn how to better filter spam messages. What is the task in this setting?

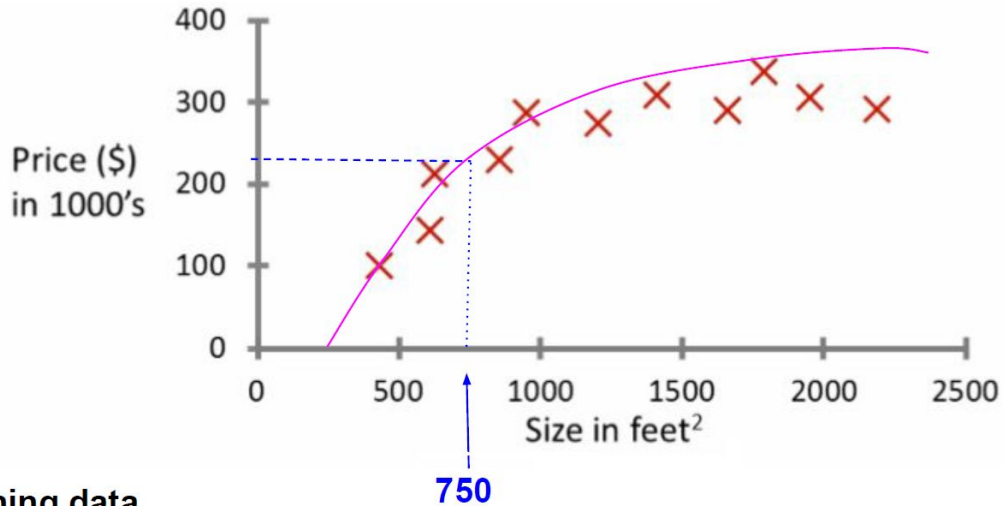
Classifying mails into the categories "spam" or "not spam"

Short Detour: Machine Learning



Supervised learning

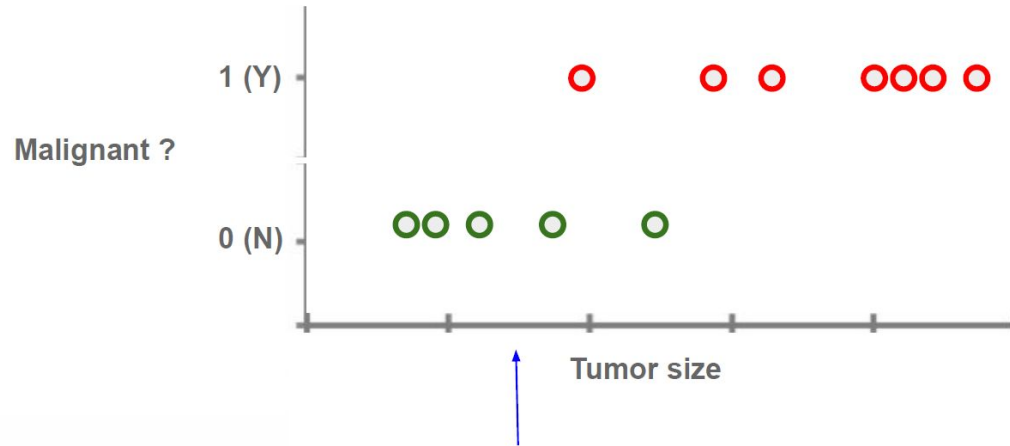
Regression predicts a continuous outcome variable



Requires training data

Supervised learning

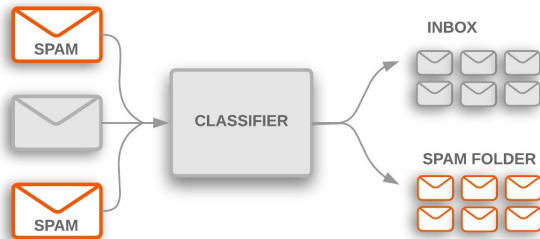
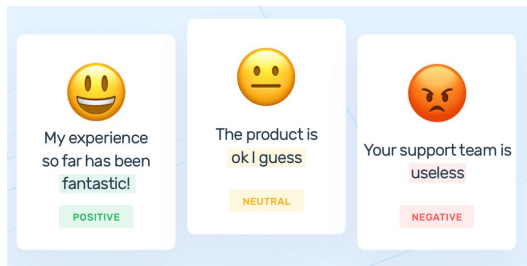
Classification predicts a discrete outcome variable



Requires training data

Text Classification

- One of the most basic and popular NLP tasks
- Many connections with what we're doing in this course
 - Tutorial 1 where we labeled whether some content is sexist or hateful
 - Exercise 1 where you do the same but with different settings
- Many connections to content analysis
- Supervised Machine Learning



Check out https://lena-voita.github.io/nlp_course/text_classification.html for an overview of different text classification datasets

Text Classification

- Given 'something' (a document, a number, a set of numbers, etc), classify it based on a **fixed** number of categories ('classes')
- Numerical things are easy to classify because computers know bits (0s and 1s)
- Therefore, we turn words to numbers
 - Obtain a **representation**
 - Classify



SORTING THINGS OUT

CLASSIFICATION AND ITS CONSEQUENCES

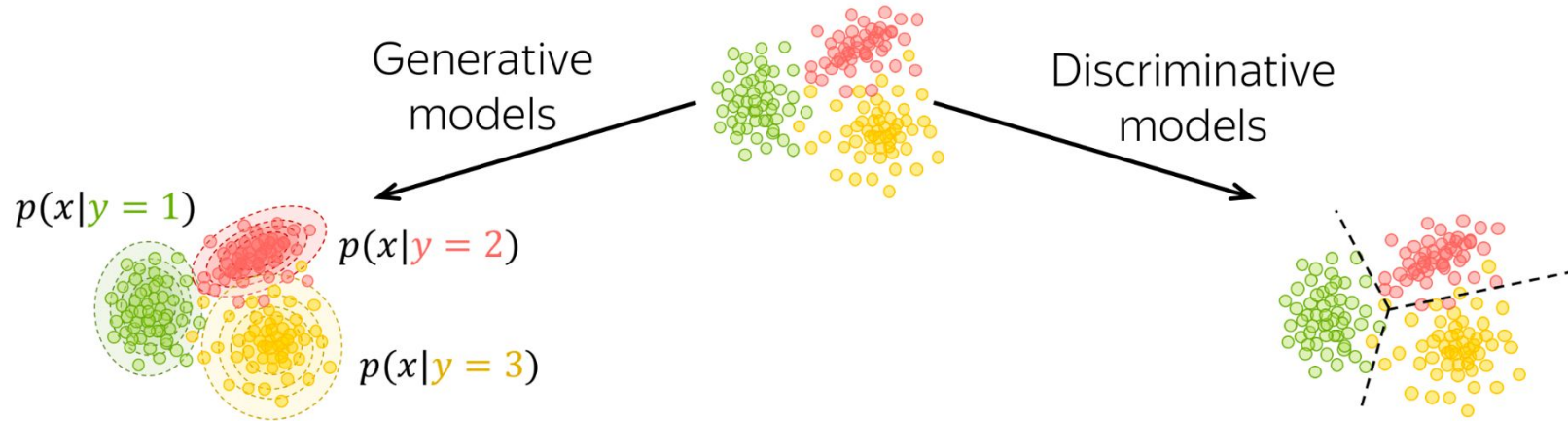
GEOFFREY C. BOWKER AND SUSAN LEIGH STAR

Copyrighted Material

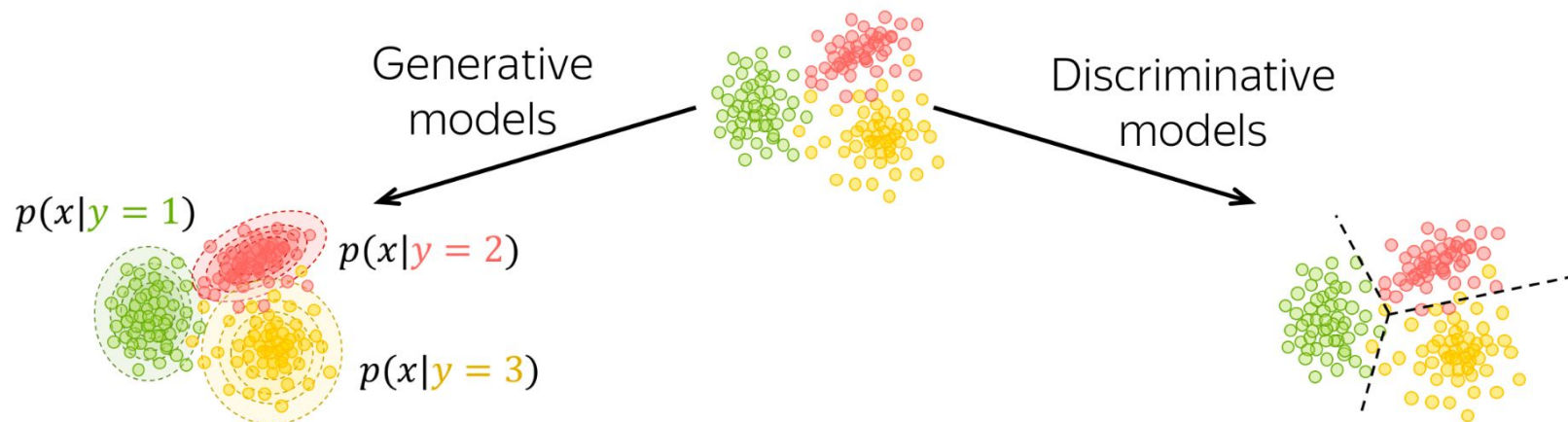
Formalizing...

- take an input x
- a fixed set of output classes $Y = \{y_1, y_2, \dots, y_M\}$
- return a predicted class $y \in Y$

Classifier



Classifier



Learn: data distribution $p(x, y) = p(x|y) \cdot p(y)$

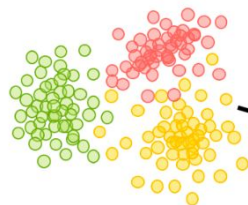
How predict: $y = \arg \max_k P(x, y = k) =$
 $= \arg \max_k P(x|y = k) \cdot P(y = k)$

Learn: boundary between classes $p(y|x)$

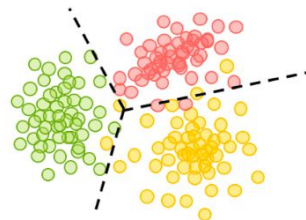
How predict: $y = \arg \max_k P(y = k|x)$

Classifier

Generative
models



Discriminative
models



$p(x|y = 1)$

$p(x|y = 2)$

$p(x|y = 3)$

Learn: data distribution $p(x, y) = p(x|y) \cdot p(y)$

How predict: $y = \arg \max_k P(x, y = k) =$
 $= \arg \max_k P(x|y = k) \cdot P(y = k)$

Joint probability distribution

Learn: boundary between classes $p(y|x)$

How predict: $y = \arg \max_k P(y = k|x)$

Conditional probability
distribution

- Lena Voita, NLP Course for you

Formalizing...

- take an input x
- a fixed set of output classes $Y = \{y_1, y_2, \dots, y_M\}$
- return a predicted class $y \in Y$

Formalizing...for NLP

- take an input x OR d (for “document”) instead of x as our input variable.
- a fixed set of output classes $Y = \{y_1, y_2, \dots, y_M\}$
- return a predicted class $y \in Y$ OR c (for “class”)

Formalizing...for NLP

- take an input x OR d (for “document”) instead of x as our input variable.
- a fixed set of output classes $Y = \{y_1, y_2, \dots, y_M\}$
- return a predicted class $y \in Y$ OR c (for “class”)
- We have a training set of N documents **labeled** with a class: $\{(d_1, c_1), \dots, (d_N, c_N)\}$.
- Our goal: learn a classifier that is capable of mapping from a new document d to its correct class $c \in C$, where C is some set of useful document classes.
- The correct class is $\operatorname{argmax}_{c \in C} P(c|d)$, i.e., which has the maximum conditional probability given the document.

Formalizing...for NLP

- $c' = \operatorname{argmax}_{c \in C} P(c|d)$, where c' is our estimated label of the new document d

Bayes Rule

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

- $c' = \operatorname{argmax}_{c \in C} P(c|d) = \operatorname{argmax}_{c \in C} (P(d|c) * P(c))/P(d)$ [Using **Bayes Rule**]
-
- $c' = \operatorname{argmax}_{c \in C} (P(f_1, f_2, \dots, f_n|c) * P(c))/P(d)$ [Representing d as a set of items $(f_1 \dots f_n)$]

Formalizing...for NLP

- $P(f_1, f_2, \dots, f_n | c) = P(f_1 | c) \cdot P(f_2 | c) \cdot \dots \cdot P(f_n | c)$

$$c' = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f | c)$$

Formalizing...for NLP

- $P(f_1, f_2, \dots, f_n | c) = P(f_1 | c) \cdot P(f_2 | c) \cdot \dots \cdot P(f_n | c)$

$$c' = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f | c)$$

$$N_c / N_{doc}$$

what percentage of the
documents in our training set
(*doc*) are in class *c*.

Formalizing...for NLP

- $P(f_1, f_2, \dots, f_n | c) = P(f_1 | c) \cdot P(f_2 | c) \cdot \dots \cdot P(f_n | c)$

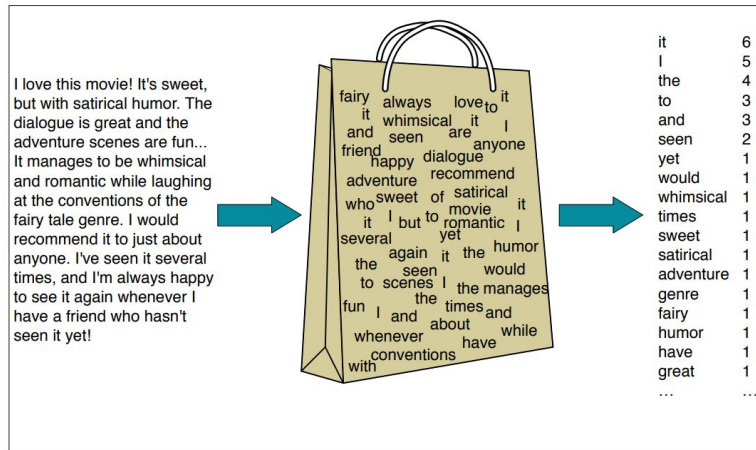
$$c' = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f | c)$$

$$N_c / N_{doc}$$

what percentage of the documents in our training set (*doc*) are in class *c*.

Obtaining Numerical Representations

- The most basic: Bag-of-words (BoW)
- Bag of Words assumption: word order does not matter



Jurafsky and Martin, Speech and Language Processing

Obtaining Numerical Representations

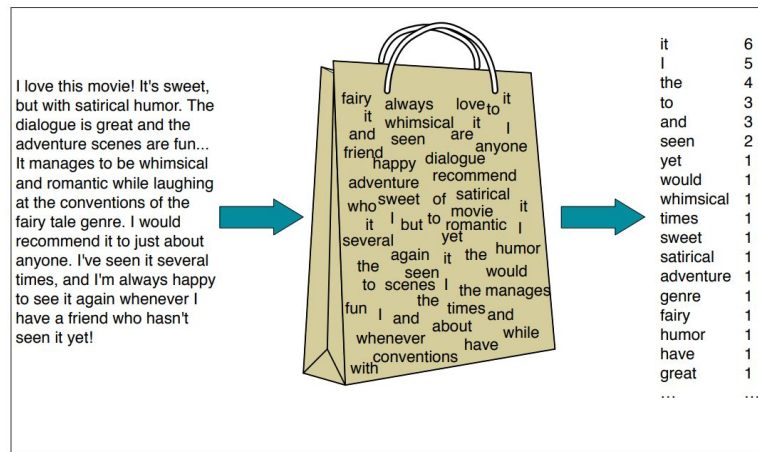
- The most basic: Bag-of-words (BoW)
- Bag of Words assumption: word order does not matter



When can this assumption be problematic?

- $P(f_1, f_2, \dots, f_n | c) = P(f_1 | c) \cdot P(f_2 | c) \cdot \dots \cdot P(f_n | c)$

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



Jurafsky and Martin, Speech and Language Processing

Formalizing...for NLP

- $P(w_1, w_2, \dots, w_n | c) = P(w_1 | c) \cdot P(w_2 | c) \cdot \dots \cdot P(w_n | c)$

$$c' = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(w | c)$$

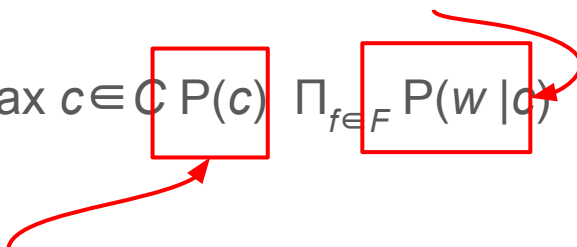
$$N_c / N_{doc}$$

what percentage of the
documents in our training set
(*doc*) are in class *c*.

Formalizing...for NLP

- $P(w_1, w_2, \dots, w_n | c) = P(w_1 | c) \cdot P(w_2 | c) \cdot \dots \cdot P(w_n | c)$

$$P'(w_i | c) = \text{count}(w_i, c) / \sum_{w \in V} \text{count}(w, c)$$

$$c' = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(w | c)$$


$$N_c / N_{doc}$$

what percentage of the documents in our training set (*doc*) are in class *c*.

Formalizing...for NLP

- $P(w_1, w_2, \dots, w_n | c) = P(w_1 | c) \cdot P(w_2 | c) \cdot \dots \cdot P(w_n | c)$

the fraction of times the word w_i appears among all words in

all documents of class c in a total vocabulary V

$$P'(w_i | c) = \text{count}(w_i, c) / \sum_{w \in V} \text{count}(w, c)$$

$$c' = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(w_f | c)$$

$$N_c / N_{doc}$$

what percentage of the documents in our training set (doc) are in class c .

Smoothing: zero numerators make everything zero

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

Toy Example: Sentiment Analysis

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

Let's drop 'with' because it doesn't occur in the training set

$$P(+) = ?$$

$$P(-) = ?$$

$$P(\text{'predictable'} | +) =$$

$$P(\text{'predictable'} | -) =$$

$$P(\text{'no'} | +) =$$

$$P(\text{'no'} | -) =$$

$$P(\text{'fun'} | +) =$$

$$P(\text{'fun'} | -) =$$

Toy Example: Sentiment Analysis

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

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

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

$$P(\text{'predictable'} | +) =$$

$$P(\text{'predictable'} | -) =$$

$$P(\text{'no'} | +) =$$

$$P(\text{'no'} | -) =$$

$$P(\text{'fun'} | +) =$$

$$P(\text{'fun'} | -) =$$

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

Toy Example: Sentiment Analysis

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

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

$V = 20$ (why not 23?)

$$V(+) = 9$$

$$V(-) = 14$$

$$3 + 5 + 6 + 2 + 6$$

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

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

$$P(\text{'predictable'} | +) =$$

$$P(\text{'predictable'} | -) =$$

$$P(\text{'no'} | +) =$$

$$P(\text{'no'} | -) =$$

$$P(\text{'fun'} | +) =$$

$$P(\text{'fun'} | -) =$$

Toy Example: Sentiment Analysis

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

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

$V = 20$ (why not 23?)

$$V(+)=9$$

$$V(-)=14$$

$$\text{count}(\text{'predictable'}, +) = 0, \text{count}(\text{'predictable'}, -) = 1$$

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

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

$$P(\text{'predictable'} | +) = (0 + 1) / (9 + 20)$$

$$P(\text{'predictable'} | -) = (1 + 1) / (14 + 20)$$

$$P(\text{'no'} | +) =$$

$$P(\text{'no'} | -) =$$

$$P(\text{'fun'} | +) =$$

$$P(\text{'fun'} | -) =$$

Toy Example: Sentiment Analysis

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

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

$V = 20$ (why not 23?)

$$V(+) = 9$$

$$V(-) = 14$$

$$\text{count}(\text{'no'}, +) = 0, \text{count}(\text{'no'}, -) = 1$$

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

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

$$P(\text{'predictable'} | +) = (0 + 1) / (9 + 20)$$

$$P(\text{'predictable'} | -) = (1 + 1) / (14 + 20)$$

$$P(\text{'no'} | +) = (0 + 1) / (9 + 20)$$

$$P(\text{'no'} | -) = (1 + 1) / (14 + 20)$$

$$P(\text{'fun'} | +) =$$

$$P(\text{'fun'} | -) =$$

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

Toy Example: Sentiment Analysis

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

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

$V = 20$ (why not 23?)

$$V(+) = 9$$

$$V(-) = 14$$

$$\text{count}(\text{'fun'}, +) = 1, \text{count}(\text{'fun'}, -) = 0$$

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

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

$$P(\text{'predictable'} | +) = (0 + 1)/(9 + 20)$$

$$P(\text{'predictable'} | -) = (1 + 1)/(14 + 20)$$

$$P(\text{'no'} | +) = (0 + 1)/(9 + 20)$$

$$P(\text{'no'} | -) = (1 + 1)/(14 + 20)$$

$$P(\text{'fun'} | +) = (1 + 1)/(9 + 20)$$

$$P(\text{'fun'} | -) = (0 + 1)/(14 + 20)$$

Toy Example: Sentiment Analysis

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

$$P(+).P(\text{'predictable no fun'}) = 0.4 * (1 * 1 * 2)/29^3 = 3.2 * 10^{-5}$$

$$P(-).P(\text{'predictable no fun'}) = 0.6 * (2 * 2 * 1)/34^3 = 6.1 * 10^{-5}$$

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

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

$$P(\text{'predictable'} | +) = (0 + 1)/(9 + 20)$$

$$P(\text{'predictable'} | -) = (1 + 1)/(14 + 20)$$

$$P(\text{'no'} | +) = (0 + 1)/(9 + 20)$$

$$P(\text{'no'} | -) = (1 + 1)/(14 + 20)$$

$$P(\text{'fun'} | +) = (1 + 1)/(9 + 20)$$

$$P(\text{'fun'} | -) = (0 + 1)/(14 + 20)$$

Toy Example: Sentiment Analysis

$$P'(w_i | c) = (\text{count}(w_i, c) + 1) / \sum_{w \in V} (\text{count}(w, c) + 1)$$

| | Cat | Documents |
|----------|-----|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable <input type="text"/> no fun |

$$P(+).P(\text{'predictable no fun'}) = 0.4 * (1 * 1 * 2)/29^3 = 3.2 * 10^{-5}$$

$$P(-).P(\text{'predictable no fun'}) = 0.6 * (2 * 2 * 1)/34^3 = 6.1 * 10^{-5}$$

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

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

$$P(\text{'predictable'} | +) = (0 + 1)/(9 + 20)$$

$$P(\text{'predictable'} | -) = (1 + 1)/(14 + 20)$$

$$P(\text{'no'} | +) = (0 + 1)/(9 + 20)$$

$$P(\text{'no'} | -) = (1 + 1)/(14 + 20)$$

$$P(\text{'fun'} | +) = (1 + 1)/(9 + 20)$$

$$P(\text{'fun'} | -) = (0 + 1)/(14 + 20)$$

Evaluating Text Classification

Sources: [4][5][6][7][8][9][10][11][12] view · talk · edit

| | | Predicted condition | | | |
|---|-----------------------------|---|---|---|---|
| | | Positive (PP) | Negative (PN) | Informedness, bookmaker informedness (BM) = TPR + TNR − 1 | Prevalence threshold (PT) = $\frac{\sqrt{\text{TPR} \times \text{FPR}} - \text{FPR}}{\text{TPR} - \text{FPR}}$ |
| Actual condition | Total population = P + N | | | | |
| | Positive (P) | True positive (TP), hit | False negative (FN), type II error, miss, underestimation | True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power = $\frac{\text{TP}}{P} = 1 - \text{FNR}$ | False negative rate (FNR), miss rate = $\frac{\text{FN}}{P} = 1 - \text{TPR}$ |
| | Negative (N) | False positive (FP), type I error, false alarm, overestimation | True negative (TN), correct rejection | False positive rate (FPR), probability of false alarm, fall-out = $\frac{\text{FP}}{N} = 1 - \text{TNR}$ | True negative rate (TNR), specificity (SPC), selectivity = $\frac{\text{TN}}{N} = 1 - \text{FPR}$ |
| Prevalence = $\frac{P}{P + N}$ | | Positive predictive value (PPV), precision = $\frac{\text{TP}}{\text{PP}} = 1 - \text{FDR}$ | False omission rate (FOR) = $\frac{\text{FN}}{\text{PN}} = 1 - \text{NPV}$ | Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$ | Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$ |
| Accuracy (ACC) = $\frac{\text{TP} + \text{TN}}{P + N}$ | | False discovery rate (FDR) = $\frac{\text{FP}}{\text{PP}} = 1 - \text{PPV}$ | Negative predictive value (NPV) = $\frac{\text{TN}}{\text{PN}}$ = 1 − FOR | Markedness (MK), deltaP (Δp) = PPV + NPV − 1 | Diagnostic odds ratio (DOR) = $\frac{\text{LR}^+}{\text{LR}^-}$ |
| Balanced accuracy (BA) = $\frac{\text{TPR} + \text{TNR}}{2}$ | | F ₁ score = $\frac{2\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$ | Fowlkes–Mallows index (FM) = $\sqrt{\text{PPV} \times \text{TPR}}$ | Matthews correlation coefficient (MCC) = $\frac{\sqrt{\text{TPR} \times \text{TNR} \times \text{PPV} \times \text{NPV}}}{\sqrt{\text{FNR} \times \text{FPR} \times \text{FOR} \times \text{FDR}}}$ | Threat score (TS), critical success index (CSI), Jaccard index = $\frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}$ |

Evaluating Text Classification

Sources: [4][5][6][7][8][9][10][11][12] view · talk · edit

| | | Predicted condition | | | |
|------------------|---|---|---|---|---|
| | | Positive (PP) | Negative (PN) | Informedness, bookmaker informedness (BM) $= \text{TPR} + \text{TNR} - 1$ | Prevalence threshold (PT) $= \frac{\sqrt{\text{TPR} \times \text{FPR}} - \text{FPR}}{\text{TPR} - \text{FPR}}$ |
| Actual condition | Total population $= P + N$ | | | | |
| | Positive (P) | True positive (TP), hit | False negative (FN), type II error, miss, underestimation | True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{\text{TP}}{P} = 1 - \text{FNR}$ | False negative rate (FNR), miss rate $= \frac{\text{FN}}{P} = 1 - \text{TPR}$ |
| | Negative (N) | False positive (FP), type I error, false alarm, overestimation | True negative (TN), correct rejection | False positive rate (FPR), probability of false alarm, fall-out $= \frac{\text{FP}}{N} = 1 - \text{TNR}$ | True negative rate (TNR), specificity (SPC), selectivity $= \frac{\text{TN}}{N} = 1 - \text{FPR}$ |
| | Prevalence $= \frac{P}{P + N}$ | Positive predictive value (PPV), precision $= \frac{\text{TP}}{\text{PP}} = 1 - \text{FDR}$ | False omission rate (FOR) $= \frac{\text{FN}}{\text{PN}} = 1 - \text{NPV}$ | Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$ | Negative likelihood ratio (LR-) $= \frac{\text{FNR}}{\text{TNR}}$ |
| | Accuracy (ACC) $= \frac{\text{TP} + \text{TN}}{P + N}$ | False discovery rate (FDR) $= \frac{\text{FP}}{\text{PP}} = 1 - \text{PPV}$ | Negative predictive value (NPV) $= \frac{\text{TN}}{\text{PN}}$ $= 1 - \text{FOR}$ | Markedness (MK), deltaP (Δp) $= \text{PPV} + \text{NPV} - 1$ | Diagnostic odds ratio (DOR) $= \frac{\text{LR}^+}{\text{LR}^-}$ |
| | Balanced accuracy (BA) $= \frac{\text{TPR} + \text{TNR}}{2}$ | F ₁ score $= \frac{2\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$ | Fowlkes–Mallows index (FM) $= \sqrt{\text{PPV} \times \text{TPR}}$ | Matthews correlation coefficient (MCC) $= \frac{\sqrt{\text{TPR} \times \text{TNR} \times \text{PPV} \times \text{NPV}}}{\sqrt{\text{FNR} \times \text{FPR} \times \text{FOR} \times \text{FDR}}}$ | Threat score (TS), critical success index (CSI), Jaccard index $= \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}$ |

Is any of this still relevant?

- For smaller corpora training a basic classifier can still be more useful than a deep learning model
- Fewer samples \rightarrow fewer parameters for model \rightarrow less overfitting
- How does this relate to the previous ‘overparameterization’ lecture?

On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes

| | |
|--|--|
| Andrew Y. Ng Computer Science Division University of California, Berkeley Berkeley, CA 94720 | Michael I. Jordan C.S. Div. & Dept. of Stat. University of California, Berkeley Berkeley, CA 94720 |
|--|--|

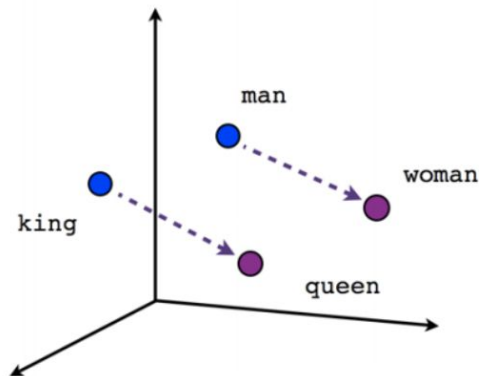
Abstract

We compare discriminative and generative learning as typified by logistic regression and naive Bayes. We show, contrary to a widely-held belief that discriminative classifiers are almost always to be preferred, that there can often be two distinct regimes of performance as the training set size is increased, one in which each algorithm does better. This stems from the observation—which is borne out in repeated experiments—that while discriminative learning has lower asymptotic error, a generative classifier may also approach its (higher) asymptotic error much faster.

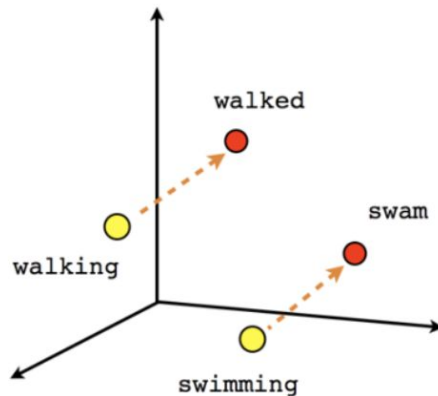
1 Introduction

Generative classifiers learn a model of the joint probability, $p(x, y)$, of the inputs x and the label y , and make their predictions by using Bayes rules to calculate $p(y|x)$, and then picking the most likely label y . Discriminative classifiers model the pos-

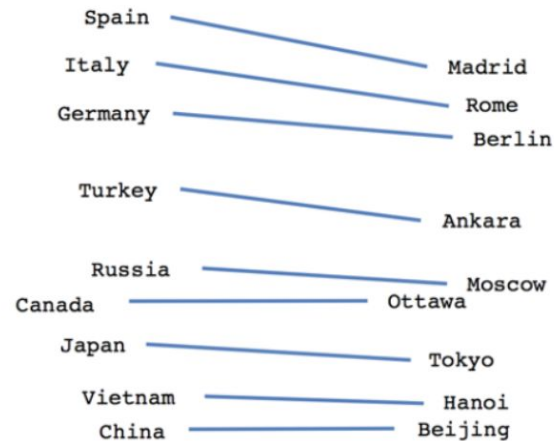
Vector Semantics



Male-Female



Verb tense



Country-Capital

Vector Semantics

“ As Wittgenstein says, ‘the meaning of words lies in their use.’ The day-to-day practice of playing language games recognizes customs and rules. It follows that a text in such established usage may contain sentences such as ‘Don’t be such an ass!’, ‘You silly ass!’, ‘What an ass he is!’ In these examples, the word ass is in familiar and habitual company, commonly collocated with you silly—, he is a silly—, don’t be such an—. **You shall know a word by the company it keeps!** ”

- John Rupert Firth, “A Synopsis of Linguistic Theory” 1957

lemma and wordform

- **A lemma or citation form**

- Same stem, part of speech, rough semantics

- **A wordform**

- The “inflected” word as it appears in text

| Wordform | Lemma |
|----------|--------|
| banks | bank |
| sung | sing |
| duermes | dormir |

Lemmas have senses

- One lemma “bank” can have many meanings:

Sense 1: ■ ...a **bank** can hold the investments in a custodial account...

Sense 2: ■ “...as agriculture burgeons on the east **bank** the river will shrink
even more”

2

- **Sense (or word sense)**
 - A discrete representation
of an aspect of a word’s meaning.
- The lemma **bank** here has two senses

Homonymy

Homonyms: words that share a form but have unrelated, distinct meanings:

- **bank₁**: financial institution, **bank₂**: sloping land
- **bat₁**: club for hitting a ball, **bat₂**: nocturnal flying mammal

2. Homographs (bank/bank, bat/bat)

3. Homophones:

1. **Write** and **right**
2. **Piece** and **peace**

Homonymy causes problems for NLP applications

- Information retrieval
 - “bat care”
- Machine Translation
 - bat: **murciélago** (animal) or **bate** (for baseball)
- Text-to-Speech
 - bass (stringed instrument) vs. bass (fish)

Polysemy

- 1. The **bank** was constructed in 1875 out of local red brick.
- 2. I withdrew the money from the **bank**
- Are those the same sense?
 - Sense 2: “A financial institution”
 - Sense 1: “The building belonging to a financial institution”
- A **polysemous** word has **related** meanings
 - Most non-rare words have multiple meanings

Synonyms

- Word that have the same meaning in some or all contexts.
 - filbert / hazelnut
 - couch / sofa
 - big / large
 - automobile / car
 - vomit / throw up
 - Water / H₂O
- Two lexemes are synonyms
 - if they can be substituted for each other in all situations
 - If so they have the same **propositional meaning**

Synonyms

- But there are few (or no) examples of perfect synonymy.
 - Even if many aspects of meaning are identical
 - Still may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.
- Example:
 - Water/H₂O
 - Big/large
 - Brave/courageous

Synonymy is a relation between senses rather than words

- Consider the words *big* and *large*
- Are they synonyms?
 - How **big** is that plane?
 - Would I be flying on a **large** or small plane?
- How about here:
 - Miss Nelson became a kind of **big** sister to Benjamin.
 - ?Miss Nelson became a kind of **large** sister to Benjamin.
- Why?
 - *big* has a sense that means being older, or grown up
 - *large* lacks this sense

Handcrafted Resources: WordNet 3.0

- A hierarchically organized lexical database
- On-line thesaurus + aspects of a dictionary
- Some [other languages](#) available or under development

- (Arabic, Finnish, German, Portuguese...)

| Category | Unique Strings |
|-----------|----------------|
| Noun | 117,798 |
| Verb | 11,529 |
| Adjective | 22,479 |
| Adverb | 4,481 |

Senses of “bass” in Wordnet

Noun

- **S: (n) bass** (the lowest part of the musical range)
- **S: (n) bass, bass part** (the lowest part in polyphonic music)
- **S: (n) bass, basso** (an adult male singer with the lowest voice)
- **S: (n) sea bass, bass** (the lean flesh of a saltwater fish of the family Serranidae)
- **S: (n) freshwater bass, bass** (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
- **S: (n) bass, bass voice, basso** (the lowest adult male singing voice)
- **S: (n) bass** (the member with the lowest range of a family of musical instruments)
- **S: (n) bass** (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Adjective

- Dan Jurafsky, <https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html>

- 84 • **S: (adj) bass, deep** (having or denoting a low vocal or instrumental range) "*a deep voice*"; "*a bass voice is lower than a baritone voice*"; "*a bass clarinet*"

How is “sense” defined in WordNet?

- The **synset (synonym set)**, the set of near-synonyms, instantiates a sense or concept, with a **gloss**
- Example: **chump** as a noun with the **gloss**:
“a person who is gullible and easy to take advantage of”
- This sense of “chump” is shared by 9 words:
chump¹, fool², gull¹, mark⁹, patsy¹, fall guy¹, sucker¹, soft touch¹, mug²
- Each of **these** senses have this same gloss
 - (Not **every** sense; sense 2 of gull is the aquatic bird)

Word Similarity

- **Synonymy**: a binary relation
 - Two words are either synonymous or not
- **Similarity (or distance)**: a looser metric
 - Two words are more similar if they share more features of meaning
- Similarity is properly a relation between **senses**
 - The word “bank” is not similar to the word “slope”
 - Bank¹ is similar to fund³
 - Bank² is similar to slope⁵
- But we’ll compute similarity over both words and senses

Why word similarity

- Information retrieval
- Question answering
- Machine translation
- Natural language generation
- Language modeling
- Automatic essay grading
- Plagiarism detection
- Document clustering

Word similarity and word relatedness

- We often distinguish **word similarity** from **word relatedness**
 - **Similar words**: near-synonyms
 - **Related words**: can be related any way
 - car, bicycle: **similar**
 - car, gasoline: **related**, not similar

Two classes of similarity algorithms

- Thesaurus-based algorithms
 - Are words “nearby” in hypernym hierarchy?
 - Do words have similar glosses (definitions)?
- Distributional algorithms
 - Do words have similar distributional contexts?

Problems with thesaurus-based meaning

- We don't have a thesaurus for every language
- Even if we do, they have problems with **recall**
 - Many words are missing
 - Most (if not all) phrases are missing
 - Some connections between senses are missing
 - Thesauri work less well for verbs, adjectives
 - Adjectives and verbs have less structured hyponymy relations

Distributional models of meaning

- Also called vector-space models of meaning
- Offer much higher recall than hand-built thesauri
 - Although they tend to have lower precision
- Zellig Harris (1954): “**oculist** and **eye-doctor** ... occur in almost the same environments.... **If A and B have almost identical environments we say that they are synonyms.**
- Firth (1957): “You shall know a word by the company it keeps!”

Further readings and materials

1. Lena Voita, 'NLP Course for you' https://lena-voita.github.io/nlp_course/
2. Jurafsky and Martin, 'Speech and Language Processing':
<https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>
3. Jacob Eisenstein, 'Natural Language Processing'
<https://cseweb.ucsd.edu/~nnakashole/teaching/eisenstein-nov18.pdf>
4. Coursera NLP Course:
<https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html>