



aivancity

SCHOOL FOR

TECHNOLOGY, BUSINESS & SOCIETY

PARIS-CACHAN

29/09/2025

Natural Language Processing (NLP)

Introduction to NLP and Text Classification

Quick word about me

- Postdoc Researcher at ISIR-CNRS (Sorbonne University)
- PhD from Paris-Saclay University (LISN-CNRS lab)
- Research topic: Multilingual Large Language Models
- More about me: <https://paullerner.github.io>
- Contact: lerner@isir.upmc.fr

Acknowledgements

This class directly builds upon:

- **Jurafsky, D., & Martin, J. H.** (2024). *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd éd.).
- **Eisenstein, J.** (2019). *Natural Language Processing*. 587.
- **Yejin Choi.** (Winter 2024). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Noah Smith.** (Winter 2023). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Benoît Sagot.** (2023-2024). *Apprendre les langues aux machines* (Collège de France)
- **Chris Manning.** (Spring 2024). Stanford CS224N: Natural Language Processing with Deep Learning
- Classes where I was/am Teacher Assistant:
 - **Christopher Kermorvant.** Machine Learning for Natural Language Processing (ENSAE)
 - **François Landes** and **Kim Gerdes.** Introduction to Machine Learning and NLP (Paris-Saclay)

Also inspired by:

- My PhD thesis: *Répondre aux questions visuelles à propos d'entités nommées* (2023)
- **Noah Smith** (2023): Introduction to Sequence Models (LxMLS)
- **Kyunghyun Cho:** Transformers and Large Pretrained Models (LxMLS 2023), Neural Machine Translation (ALPS 2021)
- My former PhD advisors **Olivier Ferret** and **Camille Guinaudeau** and postdoc advisor **François Yvon**
- My former colleagues at LISN

Program for this semester

- This morning (September 29th): What is NLP? What is a word? How can you classify a text made of words?
- Then: Practical Work 1
- Next week (October 9th): How do you get a sense of a word?
- Then: Practical Work 2
- October 23rd: N-gram and Neural Language Models
- Then: Practical Work 1/2 extra
- November 4th: Practical Work 3 (2 sessions)

Program for this semester

- November 10th: Neural Network architectures used in Large Language Models:
 - Recurrent Neural Networks
 - Attention Mechanism
- Then: Transformers pretraining and fine-tuning
- November 20th: Practical Work 4 (2 sessions)
- November 27th: Practical Work 5 (2 sessions)

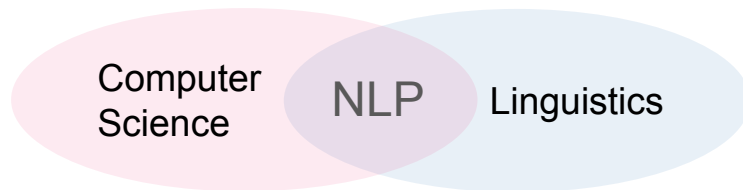
Program for this semester

- December 4th:
 - Ethical, social, and environmental issues
 - alignment: reinforcement learning from human feedback (RLHF)
 - other things happening in NLP research at the moment
- Then: Practical Work 6
- Practical Works → individual evaluation 50%
- Individual Final sitting Exam 50%

Background

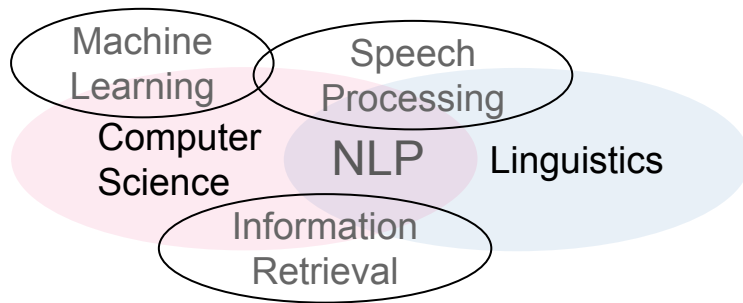
- Assumes basics in Machine Learning, Python, and Linear Algebra
 - Good exercises /cheatsheet "Day 0: Basic Tutorials" of http://lxmls.it.pt/2023/documents/LxMLS_guide_2023.pdf
- Recommended readings:
 - Daniel Jurafsky and James H. Martin. 2025. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. Online manuscript released January 12, 2025. <https://web.stanford.edu/~jurafsky/slp3>
 - Eisenstein, J. (2019). Natural Language Processing. <https://nlp.cs.princeton.edu/cos484-sp21/readings/eisenstein-nlp-notes.pdf>

Natural Language Processing (NLP)



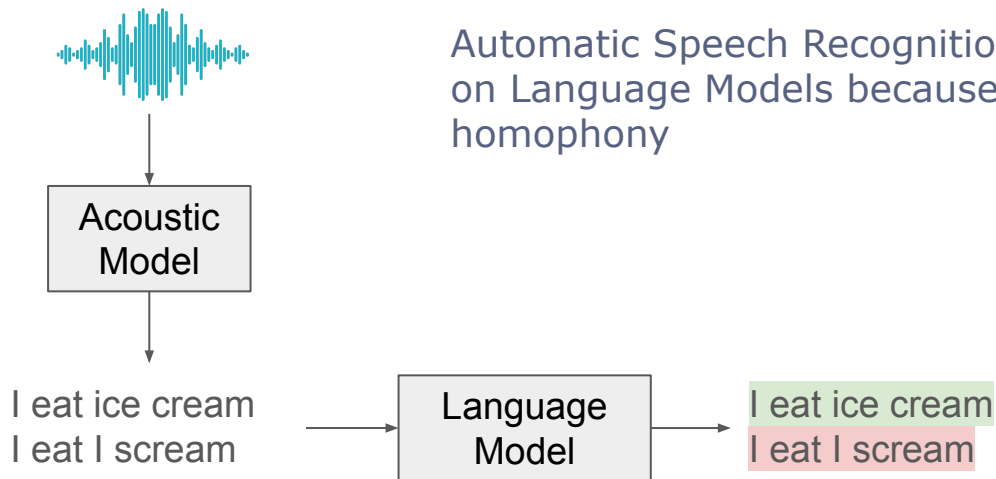
- Intersection of Computer Science and Linguistics:
 - Distributional Semantics: sense of a word from its context (next class)
 - Computational Linguistics, Computational Morphology, etc.:
study of humans: how do we speak? how do we organize lexicon?

Natural Language Processing (NLP)



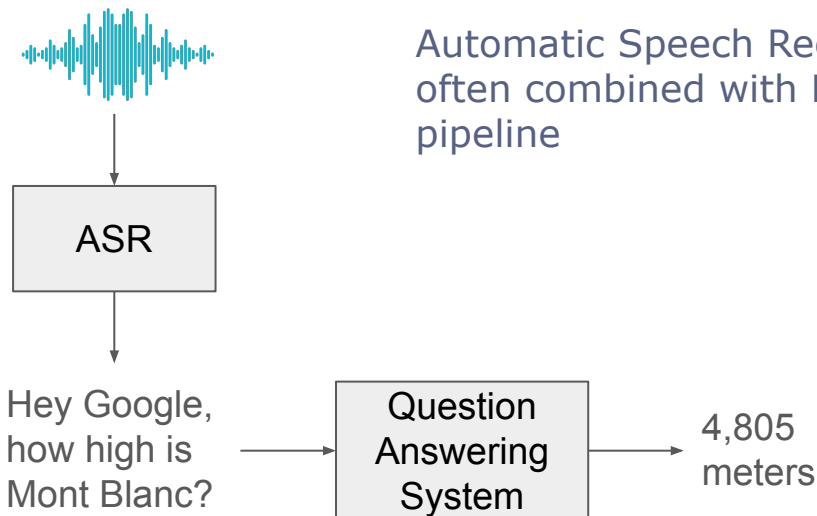
- Close to Speech Processing (Automatic Speech Recognition etc.)
- Close to Information Retrieval (Search engines like Google)
- Driven by Statistical/Machine Learning methods since the 90s
- Driven by Deep Learning since 2010s

Speech and Language Processing



Automatic Speech Recognition (ASR) relies on Language Models because of homophony

Speech and Language Processing



Automatic Speech Recognition (ASR) is often combined with NLP methods in a pipeline

Speech and Language Processing

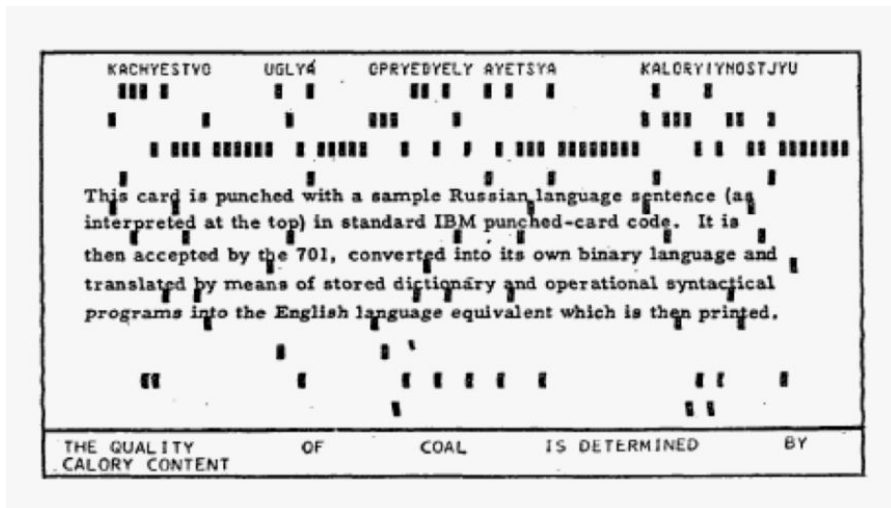


Recently moving
towards integrated,
multimodal
end-to-end models

What is scientific research?

- General goal: Pushing the limits of our **knowledge**
- **Incrementally!** Find a limit/caveat in existing method and solve it!
- For example: lack of parallelization in Recurrent Neural Networks → **Transformers** (Vaswani et al. 2017)
- "Vaswani et al. 2017": a single publication that was **submitted** to a conference, **reviewed** by scientists, then **reproduced: research != science**
- Most of methods in this class were published less than 15 years ago

NLP applications: Machine Translation



Georgetown-IBM experiment 1954

- Machine Translation is the first NLP application
- Google Translate supports 243 languages

Google Cloud Overview Solutions Products Pricing Resources

Cloud Translation

Model	Method	Usage	Price per unit
NMT	Text translations, which includes: - Language detection - Text translation - Batch text translation - XLIX document translation - Romanize text	First 500,000 characters per month	Free (up to \$10 credit every month)
		Over 500,000 characters per month	\$20 per million characters*
		Over 1 billion characters per month	We recommend that you contact a sales representative to discuss discount pricing.
	Document translation (DOCX, PPT, and PDF formats only)	Pages sent to the API per month	\$0.08 per page†

PRICING

Cloud Translation pricing

Pricing examples

Charged characters

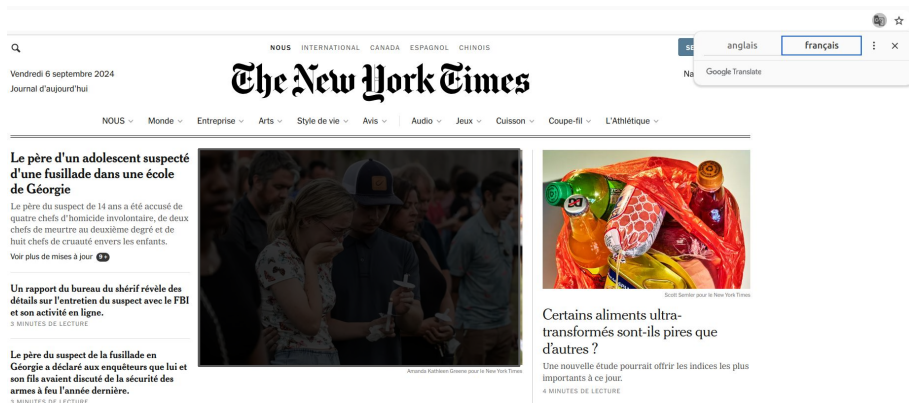
Charged projects

Other Google Cloud costs

What's next

NLP applications: Machine Translation

Ubiquitous on the web and social media



The screenshot shows the front page of The New York Times. A Google Translate overlay is visible in the top right corner, showing the text 'anglais' and 'français' with a dropdown menu. The main headline is 'Le père d'un adolescent suspecté d'une fusillade dans une école de Géorgie'. Below it, there is a sub-headline 'Un rapport du bureau du shérif révèle des détails sur l'entretien du suspect avec le FBI et son activité en ligne.' and another sub-headline 'Le père du suspect de la fusillade en Géorgie a déclaré aux enquêteurs que lui et son fils avaient discuté de la sécurité des armes à feu l'année dernière.' To the right of the main headline, there is a photo of a crowd of people and a headline 'Certains aliments ultra-transformés sont-ils pires que d'autres ?'.

Desmond Elliott
@delliott

Fun new paper led by [@IngoZiegler](#) and [@akoksal](#) that shows how we can use retrieval augmentation to create high-quality supervised fine tuning data. All you need to do is write a few examples that demonstrate the task.

[À l'origine en anglais et traduit par Google](#)

Un nouvel article intéressant dirigé par [@IngoZiegler](#) et [@akoksal](#) qui montre comment nous pouvons utiliser l'augmentation de la récupération pour créer des données de réglage fin supervisées de haute qualité. Tout ce que vous avez à faire est d'écrire quelques exemples qui illustrent la tâche.

Is this spam?

Dear Paul

As per our discussions, I hereby confirm your teaching of the following courses for the 2526 academic year:

1. Natural Language Processing: PGE4 INI for S1 (24 hours)
2. Natural Language Processing: PGE4 ALT for S2 (24 hours)

Best regards,
Anuradha

Paul Lerner – September 2025

LIDL™
Offre Exclusive – Clients Sélectionnés

Coffret Makita pour

Félicitations ! Vous avez été **sélectionné(e)** pour recevoir un **Coffret Makita Professionnel** de 108 pièces.

Cette offre est **strictement limitée** aux clients tirés au sort. Profitez-en dès maintenant avant la date limite.

ID Coupon : MAK845652

Date limite : 8-30-2025



Je réclame mon Coffret

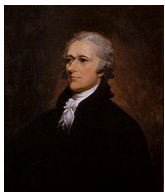
aivancity
PARIS-CACHAN

Who wrote which Federalist Papers?

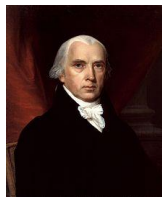
1787-8: essays anonymously written by:

Alexander Hamilton, James Madison, and John Jay
to convince New York to ratify U.S Constitution

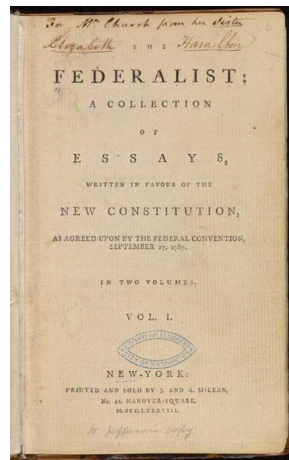
Authorship of 12 of the letters unclear between:



Alexander Hamilton



James Madison



1963: solved by Mosteller and Wallace using Bayesian methods

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

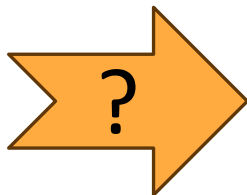


It was pathetic. The worst part about it was the boxing scenes.

What is the subject of this article?

MEDLINE Article

MeSH Subject Category Hierarchy



Antagonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy

Embryology

Epidemiology

...

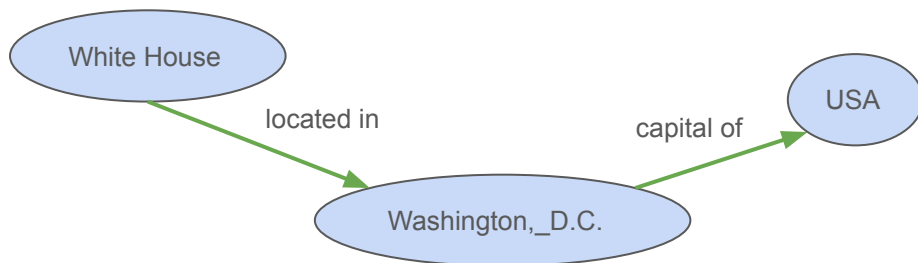
Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification (who wrote this?)
- Language Identification (is this Portuguese?)
- Sentiment analysis

NLP applications: Information Extraction

[Washington, D.C.](#) != [George Washington](#)

Washington is the capital of the USA. It hosts the White House.



- From unstructured text to knowledge graphs
- Named Entity Recognition
- Named Entity Disambiguation
- Coreference resolution
- Relation Extraction

NLP applications: Information Extraction



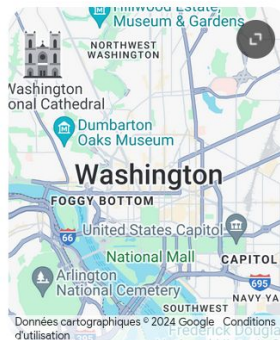
Tightly connected with
Information Retrieval

Tous Images Actualités Vidéos Livres Web Finance

Outils

Washington

Capitale des États-Unis



Météo

sam. dim. lun.
☁ 23° ☀ 23° ☀ 26°
weather.com

Y aller

✈ 8h 25min
à partir de Paris

NLP applications: Information Extraction

Not only for advancing human knowledge

This **LOAN AGREEMENT**, dated as of **November 17, 2014** (this “**Agreement**”), is made by and among **Auxilium Pharmaceuticals, Inc.**, a corporation incorporated under the laws of the State of **Delaware** (“**U.S. Borrower**”), **Auxilium UK LTD**, a private company limited by shares registered in **England and Wales** (“**UK Borrower**”), **Endo Pharmaceuticals Inc.**, a corporation incorporated under the laws of the State of **Delaware** (“**Lender**”).

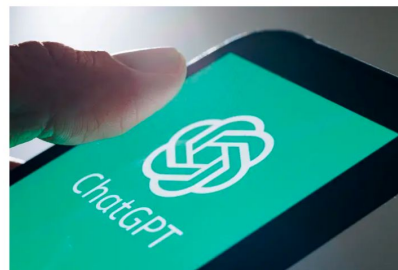
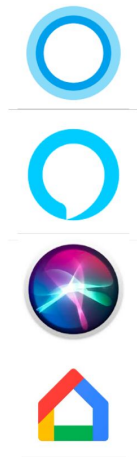
Page	
SHAREHOLDER INFORMATION	
THE WALT DISNEY CO.	
DISCS CORPORATE ACTION CALENDAR	
LATEST PUBLIC OFFERING	
Date of offering	1/83
Shares offered	1.00M
Share Price	\$ 66.88
Lead Manager	Morgan Stanley
Type	Common Stock
Split Adj:	1.39
INSIDER TRADING	
Net \$ Value Buys and Sells As Of 01/15/05	773
(1985 - Present In Dollars)	# of Sellers 788
Lowest activity	12/97 -372.00MLN
Highest activity	08/02 10.17MLN
Mean:	-15.95MLN
Most recent 45 days	.00
INSTITUTIONAL OWNERSHIP	
# of Buyers	773
# of Sellers	788
# of Holders	1,970
Shares Held	1.39BLN
% Shares Out.	68.12
Shares Purchased	1.75MLN
Australia 61 2 9777 0600	Brazil 5511 3048 4500
Hong Kong 852 2577 6000	Japan 81 3 3201 6900
	Singapore 65 6212 1000 U.S. 1 212 318 2000
	Copyright 2005 Bloomberg L.P.
	1 24-Jun-05 15:16:57

LOAN AGREEMENT									
[HELP] for explanation. Enter *GD* to select aggregate portfolio and see detailed information 002723178154-000									
DIS US HOLDINGS SEARCH									
THE WALT DISNEY CO. Page 1 / 100									
Holder name	Portfolio Name	Source	Held	Percent	Latest Filing				
BARCLAYS GLOBAL	BARCLAYS BANK PLC	13F	91,334M	4.470	7,912M	09/04			
ACITIGROUP INCORP	CITIGROUP INCORPORAT	13F	71,012M	3.475	893,816	09/04			
STATE STREET	STATE STREET CORPORA	13F	69,238M	3.369	1,214M	09/04			
FIDELITY MANAGEN	FIDELITY MANAGEMENT	13F	67,611M	3.305	4,988M	09/04			
SOUTHERNSTEN ASST	SOUTHERNSTEN ASSET M	13F	52,949M	2.591	3,194M	09/04			
VANGUARD GROUP	VANGUARD GROUP INC	13F	43,710M	2.139	973,055	12/04			
7ST FARM MU AUTO	STATE FARM MUTUAL AU	13F	42,234M	2.067	10,300	09/04			
MELLON BANK N A	MELLON BANK CORP	13F	39,545M	1.935	2,988M	09/04			
WORLD ABBETT & CO	LORD ABBETT & CO	13F	37,460M	1.832	286,434	12/04			
MORGAN STANLEY	MORGAN STANLEY	13F	31,643M	1.545	1,348M	09/04			
NORTHERN TRUST C	NORTHERN TRUST CORPO	13F	26,061M	1.275	39,493	09/04			
DEUTSCHE BANK AK	DEUTSCHE BANK AG	13F	21,990M	1.076	1,570M	09/04			
DISNEY ROY EDWAR	n/a	Form 4	17,279M	0.845	-7,726	08/03			
JANUS CAPITAL	JANUS CAPITAL CORPORA	13F	17,156M	0.840	-2,077M	09/04			
ICAPITAL RSCH MGM	CAPITAL RESEARCH AND	13F	17,142M	0.839	1,907M	09/04			
TUKMAN CAP MGMT	TUKMAN CAPITAL MANAG	13F	16,962M	0.830	1,861M	09/04			
TOME PRICE	TOME PRICE ASSOCIA	13F	16,810M	0.823	-1,367M	09/04			
Sub-totals for current page:						680,137M	33.285		
* Money market directory info available. Select portfolio, then hit IP<GD>.									
Australia 61 2 9777 0600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410 Hong Kong 852 2577 6000 Japan 81 3 3201 6900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2005 Bloomberg L.P. 1 24-Jun-05 15:13:32									

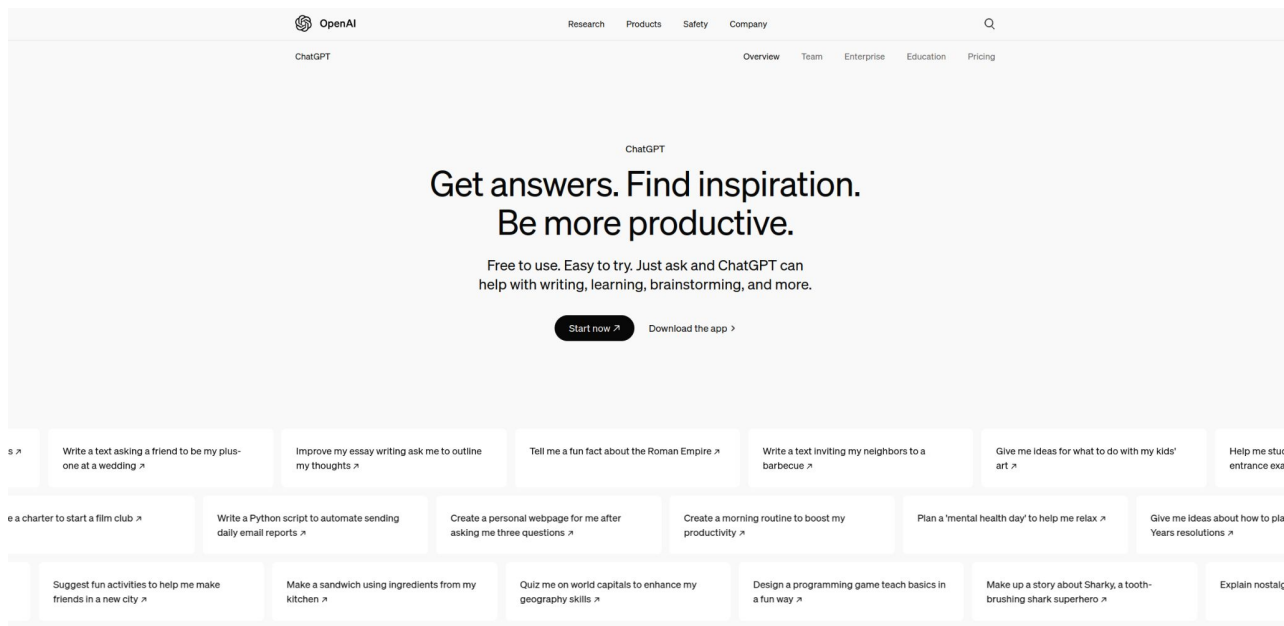
Bloomberg

NLP applications: chatbots

- Task-oriented like Siri/Alexa or chitchat like ChatGPT (OpenAI)
- Moving from a complicated pipeline (ASR, Information Extraction, Information Retrieval) to end-to-end language modeling



NLP applications: chatbots



NLP applications: chatbots: demo

32 lines of python code

```
1 from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig
2
3 def pipeline(text, model, tokenizer):
4     messages = [
5         {"role": "user", "content": text}
6     ]
7     inputs = tokenizer.apply_chat_template(messages, return_dict=True, tokenize=True, return_tensors="pt", add_generation_prompt=True)
8     for k, v in inputs.items():
9         inputs[k] = v.cuda()
10    input_length = inputs["input_ids"].shape[1]
11    output = model.generate(**inputs, max_new_tokens=128)
12    output = tokenizer.batch_decode(output[:, input_length:], skip_special_tokens=True, clean_up_tokenization_spaces=True)
13    return output
14
15
16 def user_loop(model, tokenizer):
17     while True:
18         answer = input(f">>> ").strip()
19         output = pipeline(answer, model, tokenizer)[0]
20         print(f"{output}\n")
21
22
23 def main():
24     model_name = "google/gemma-2-2b-it"
25     model = AutoModelForCausalLM.from_pretrained(
26         model_name,
27         device_map="auto",
28         token=True,
29         quantization_config=BitsAndBytesConfig(load_in_4bit=True)
30     )
31     tokenizer = AutoTokenizer.from_pretrained(model_name)
32     user_loop(model, tokenizer)
```

NLP applications: chatbots: demo

```
>>> what is distributional semantics?
```

```
## Distributional Semantics: Understanding Words in Context
```

```
Distributional semantics is a branch of natural language processing (NLP) that focuses on the **relationships between words and their meaning based on their co-occurrence in a large corpus of text**.
```

```
Think of it like this: instead of focusing on individual words and their exact definitions, distributional semantics analyzes how words are used in context.
```

```
**Here's a breakdown:**
```

```
* **Words are not isolated:** They are part of a larger network of words and phrases.
```

```
* **Meaning is derived from context:** The meaning of a word is determined by how it is
```

NLP applications: chatbots: demo

```
>>> I'm teaching an NLP class at the aivancity AI school, any advices?  
That's awesome! Teaching NLP is a fantastic way to share the power of AI with the next generation. Here's some  
advice to make your class engaging and impactful:  
  
**1. Know Your Audience:**  
  
* **Beginner or Advanced?** Are you targeting absolute beginners or those with some prior knowledge? Tailor yo  
ur content and pace accordingly.  
* **AI Enthusiasts vs. General Interest:** Are you aiming for a more technical audience or those with a broader  
interest in AI?  
* **Learning Style:** Consider how your students learn best (visual, auditory, kinesthetic). Incorporate divers  
e teaching methods.
```

NLP applications: chatbots: demo

```
>>> you're speaking live to students of the aivancity AI school, introduce yourself  
Hello everyone! 🖐️ I'm Gemma, an AI assistant here to help you learn and explore the fascinating world of AI.
```

```
I'm excited to be speaking with you today at Aivancity AI School. I've been trained on a massive dataset of text and code, which allows me to understand and generate human-like text.
```

```
What are you working on today? Do you have any questions about AI concepts, or are you working on a specific project? I'm here to help!
```

```
Let's dive into the world of AI together! 🤖
```

Break for questions and "appel"

What is a word?

- Open question in phonology vs. morphology
- Inflection: is brother != brothers**s**?
- Compounding: is motorbike == motor + bike?
- Multi-word expressions: mother in law == 1 or 3 words?
- Homonymy: is chair (furniture) != chair (person)?
- Orthography: is modeling != modell**l**ing?

NLP deals with orthographic words

- "My brother is sitting on a chair" →
['My', 'brother', 'is', 'sitting', 'on', 'a', 'chair']
(**tokenization**: sequence of *tokens*)
- Inflection (brother vs brothers): usually not modeled
- Compounding (motorbike vs motor + bike): usually not modeled
- Multi-word expressions (mother in law): usually not modeled
- Homonymy (chair [furniture] vs chair [person]): usually modeled after sharing an initial representation
- **Orthography**: 'modeling' != 'mode11ing'

Issues in Tokenization

- Can't just blindly remove punctuation:
 - m.p.h., Ph.D., AT&T, cap'n
 - prices (\$45.55)
 - dates (01/02/06)
 - URLs (<http://www.stanford.edu>)
 - hashtags (#nlproc)
 - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
 - "are" in we're, French "je" in j'ai, "le" in l'honneur

Using regular expressions

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)      # set flag to allow verbose regexps
...     (?:[A-Z]\.)+        # abbreviations, e.g. U.S.A.
...     | \w+(?:-\w+)*      # words with optional internal hyphens
...     | \$?\d+(?:\.\d+)?%? # currency, percentages, e.g. $12.40, 82%
...     | \.\.\.           # ellipsis
...     | [][.,;"'()?:_'-] # these are separate tokens; includes ], [
...     ','
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Why regular expressions?

- A formal language for specifying text strings
- How can we search for mentions of these cute animals in text?
 - woodchuck
 - woodchucks
 - Woodchuck
 - Woodchucks
 - Groundhog
 - groundhogs



Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
<code>[wW]oodchuck</code>	Woodchuck, woodchuck
<code>[1234567890]</code>	Any one digit

- Ranges using the dash `[A-Z]`

Pattern	Matches	
<code>[A-Z]</code>	An upper case letter	<u>D</u> renched Blossoms
<code>[a-z]</code>	A lower case letter	<u>m</u> y beans were impatient
<code>[0-9]</code>	A single digit	Chapter <u>1</u> : Down the Rabbit Hole

Regular Expressions: More Disjunctions

- Groundhog is another name for woodchuck!
- The pipe symbol | for disjunction

Pattern	Matches
<code>groundhog woodchuck</code>	<code>woodchuck</code>
<code>yours mine</code>	<code>yours</code>
<code>a b c</code>	<code>= [abc]</code>
<code>[gG] roundhog [Ww] oodchuck</code>	<code>Woodchuck</code>

Paul Lerner – September 2025



Regular Expressions: Convenient aliases

Pattern	Expansion	Matches	Examples
<code>\d</code>	<code>[0-9]</code>	Any digit	Fahre ^{ne} it <u>4</u> 51
<code>\D</code>	<code>[^0-9]</code>	Any non-digit	<u>B</u> lue Moon
<code>\w</code>	<code>[a-zA-Z0-9_]</code>	Any alphanumeric or _	<u>D</u> aiyu
<code>\W</code>	<code>[^\w]</code>	Not alphanumeric or _	Look <u>!</u>
<code>\s</code>	<code>[\r\t\n\f]</code>	Whitespace (space, tab)	Look <u> </u> up
<code>\S</code>	<code>[^\s]</code>	Not whitespace	<u>L</u> ook up

Wildcards, optionality, repetition: . ? * +

Pattern	Matches	Examples
<code>beg.n</code>	Any char	<u>begin</u> <u>begun</u> <u>beg3n</u> <u>beg_n</u>
<code>woodchucks?</code>	Optional s	<u>woodchuck</u> <u>woodchucks</u>
<code>to*</code>	0 or more of previous char	<u>t</u> <u>to</u> <u>too</u> <u>tooo</u>
<code>to+</code>	1 or more of previous char	<u>to</u> <u>too</u> <u>tooo</u> <u>toooo</u>



Stephen C Kleene

Kleene *, Kleene +

A note about Python regular expressions

- Regex and Python both use backslash "\" for special characters. You must type extra backslashes!
 - `"\\d+"` to search for 1 or more digits
 - `"\n"` in Python means the "newline" character, not a "slash" followed by an "n". Need `"\\n"` for two characters.
- Instead: use Python's raw string notation for regex:
 - `r"[tT]he"`
 - `r"d+"` matches one or more digits

Strings of characters

- Note in python: there are no characters, only strings!
- `"foo"` is a string
- `"foo"[0]` is a string (`"f"`)
- `"foo"[0][0]` is a string (`"f"`)

The iterative process of writing regex's

- Find me all instances of the word “the” in a text.

the

Misses capitalized examples

[tT]he

Incorrectly returns other or Theology

\W[tT]he\W

- Good tool to learn <https://regex101.com/>

Tokenization in languages without spaces

- Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!
- How do we decide where the token boundaries should be?

Word tokenization in Chinese

- Chinese words are composed of characters called "hanzi" (or sometimes just "zi")
- Each one represents a meaning unit called a morpheme.
- Each word has on average 2.4 of them.
- But deciding what counts as a word is complex and not agreed upon.

How to do word tokenization in Chinese?

- 姚明进入总决赛 “Yao Ming reaches the finals”

- 3 words?

- 姚明 进入 总决赛

- YaoMing reaches finals

- 5 words?

- 姚 明 进入 总 决赛

- Yao Ming reaches overall finals

- 7 characters? (don't use words at all):

- 姚 明 进 入 总 决 赛

- Yao Ming enter enter overall decision game

Zipf's Law

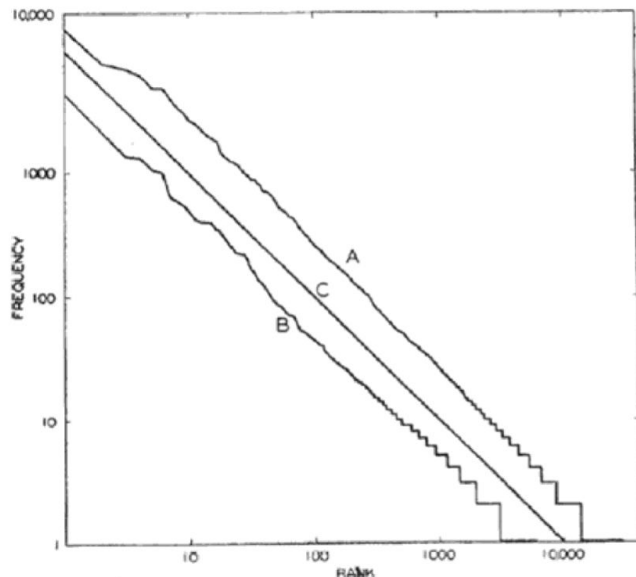


Fig. 2-1. The rank-frequency distribution of words. (A) The James Joyce data; (B) the Eldridge data; (C) ideal curve with slope of negative unity.

Zipf (1949) found that, when counting the frequencies of all words in a Joyce's Ulysses (**260,430 words**), that:

- the **10th** most frequent word occurred **2,653** times
- the **100th** **265** times, or 10 times less
- the **1000th** **26** times, or 100 times less

In other words, the frequency $f(w)$ of a word w is inversely proportional to its rank k : $f(w) \propto 1/k$

or $f(w) \times k = \text{constant}$

Zipf's Law

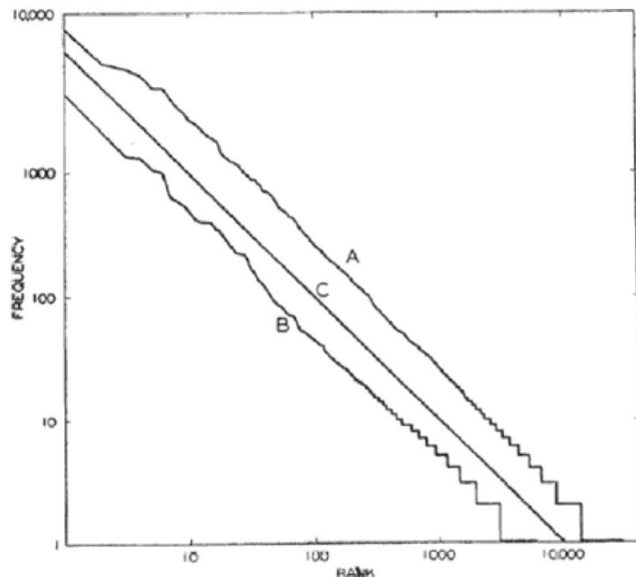
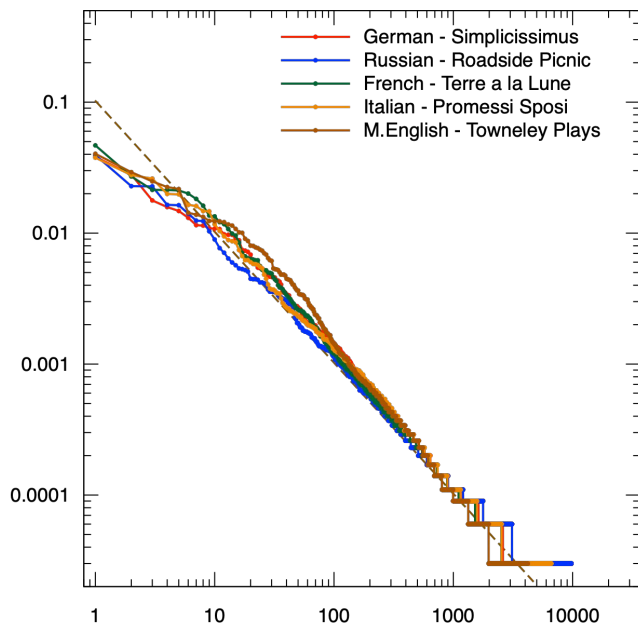


Fig. 2-1. The rank-frequency distribution of words. (A) The James Joyce data; (B) the Eldridge data; (C) ideal curve with slope of negative unity.

- The frequency $f(\mathbf{w})$ of a word $\mathbf{w}$ is inversely proportional to its rank k : $f(\mathbf{w}) \propto 1/k$
- The distribution of words frequencies have a very long tail: some words are very frequent other appear only once in the whole training corpus

Zipf's Law



- The frequency $f(w)$ of a word w is inversely proportional to its rank k : $f(w) \propto 1/k$
- The distribution of words frequencies have a very long tail: some words are very frequent other appear only once in the whole training corpus
- This empirical law was verified in most languages of the world, it's one of the canonical laws of computational linguistics

Open vs. closed vocabulary

- Training a classifier → mostly fine to have a fixed vocabulary of V *words*. Top-k ones will cover vast majority of corpus, why?
- What about for a generative model?
- For a multilingual model (e.g. English-Chinese)?
- Use a fixed vocabulary of V' *subwords* that can be combined into *words*
- "open" vocabulary but still a finite set of *tokens*

Byte Pair Encoding (BPE) token learner

- Let vocabulary be the set of all individual characters
- $= \{A, B, C, D, \dots, a, b, c, d, \dots\}$
- Repeat:
 - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
 - Add a new merged symbol 'AB' to the vocabulary
 - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
- Until k merges have been done.

BPE token learner algorithm

function BYTE-PAIR ENCODING(strings C , number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters

for $i = 1$ **to** k **do** # merge tokens til k times

$t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C

$t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating

$V \leftarrow V + t_{NEW}$ # update the vocabulary

 Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus

return V

Byte Pair Encoding (BPE) Addendum

- Most subword algorithms are run inside space-separated tokens.
- So we commonly first add a special begin-of-word symbol 'Ġ' before space in training corpus
- Next, separate into letters.

BPE token learner

Original (very fascinating 🤖) corpus:

low low low low low lowest lowest newer newer newer
newer newer newer wider wider wider new new

Add begin-of-word tokens, resulting in this vocabulary:

Ġ l o w e s t n r i d

BPE token learner

corpus

5 Ġ l o w
2 Ġ l o w e s t
6 Ġ n e w e r
3 Ġ w i d e r
2 Ġ n e w

vocabulary

Ġ l o w e s t n r i d

BPE token learner

Merge **e** **r** to **er**

corpus

5 Ġ l o w
2 Ġ l o w e s t
6 Ġ n e w e r
3 Ġ w i d e r
2 Ġ n e w

vocabulary

Ġ l o w e s t n r i d
e r

BPE token learner

Merge $\dot{G}$ n to $\dot{G}n$

corpus

5 $\dot{G}$ l o w
2 $\dot{G}$ l o w e s t
6 $\dot{G}n$ e w e r
3 $\dot{G}$ w i d e r
2 $\dot{G}n$ e w

vocabulary

$\dot{G}$ l o w e s t n r i d
e r $\dot{G}n$

BPE token learner

Merge $\dot{G}n$ e to $\dot{G}ne$

corpus

5 $\dot{G}$ l o w
2 $\dot{G}$ l o w e s t
6 $\dot{G}ne$ w er
3 $\dot{G}$ w i d er
2 $\dot{G}ne$ w

vocabulary

$\dot{G}$ l o w e s t n r i d
er $\dot{G}n$ $\dot{G}ne$

BPE token learner

The next merges are:

merge

vocabulary

(Ġne, w)	Ġ l o w e s t n r i d e r Ġn Ġne Ġnew
(Ġ, l)	Ġ l o w e s t n r i d e r Ġn Ġne Ġnew Ġl
(Ġl, o)	Ġ l o w e s t n r i d e r Ġn Ġne Ġnew Ġl Ġlo
(Ġlo, w)	Ġ l o w e s t n r i d e r Ġn Ġne Ġnew Ġl Ġlo Ġlow
(Ġnew, er)	Ġ l o w e s t n r i d e r Ġn Ġne Ġnew Ġl Ġlo Ġlow Ġnewer

BPE token segmenter algorithm

- On the test data, run each merge learned from the training data:
 - Greedily
 - In the order we learned them
 - (test frequencies don't play a role)
- So: merge every $e\ r$ to er , then merge $\dot{G}\ n$ to $\dot{G}n$, etc.
- Result:
 - Test set " $\dot{G}\ n\ e\ w\ e\ r$ " would be tokenized as a full word
 - Test set " $\dot{G}\ l\ o\ w\ e\ r$ " would be two tokens: " $\dot{G}low\ er$ "

Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification (who wrote this?)
- Language Identification (is this Portuguese?)
- Sentiment analysis

Text Classification: definition

Input:

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

Output: a predicted class $c \in C$

Classification Method: Supervised Machine Learning

Input:

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

Output:

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

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

Bayes' Rule

- For a document d and a class c

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

Naive Bayes Classifier (I)

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

MAP is “maximum a posteriori” = most likely class

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

Bayes Rule

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

Dropping the denominator

Naive Bayes Classifier (II)

"Likelihood"

"Prior"

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

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

Document d
represented as
features $x_1 \dots x_n$

Naive Bayes Classifier (III)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n | c) P(c)$$

$O(|X|^n \cdot |C|)$ parameters

Could only be estimated if a very, very large number of training examples was available.

How often does this class occur?

We can just count the relative frequencies in a corpus

Multinomial Naive Bayes Independence Assumptions

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

Bag of Words/Naive Bayes assumption: Assume position doesn't matter

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

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

The Bag of Words Representation

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
...	...

Applying Multinomial Naive Bayes Classifiers to Text Classification

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

Problems with multiplying lots of probs

There's a problem with this:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

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!

We actually do everything in log space

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

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

Note: Taking log doesn't change the ranking of classes!

The class with highest probability also has highest log probability!

Learning the Multinomial Naive Bayes Model

First attempt: maximum likelihood estimates

- simply use the frequencies in the data
- See Eisenstein for proof that it maximizes likelihood

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

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

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

Problem with Maximum Likelihood

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

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

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

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

Laplace (add-1) smoothing for Naïve Bayes

$$\begin{aligned}\hat{P}(w_i | c) &= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} \\ &= \frac{\text{count}(w_i, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V|}\end{aligned}$$

Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

Calculate $P(c_j)$ terms

- For each c_j in C do

$docs_j \leftarrow$ all docs with class $= c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k | c_j)$ terms
 - $Text_j \leftarrow$ single doc containing all $docs_j$
 - For each word w_k in *Vocabulary*

$$n_k \leftarrow \text{\# of occurrences of } w_k \text{ in } Text_j$$

$$P(w_k | c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Bias-variance tradeoff

- If α is large $\rightarrow$ uniform distribution $\rightarrow$ underfit training data

- high bias
- low variance

- if α is small $\rightarrow$ overfit training data

- high variance
- low bias

Calculate $P(w_k | c_j)$ terms

- $Text_j \leftarrow$ single doc containing all $docs_j$
- For each word w_k in *Vocabulary*
 $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k | c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha | \text{Vocabulary} |}$$

Let's do a worked sentiment example!

	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 with no fun

A worked sentiment example with add-1 smoothing

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 with no fun

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}} \quad P(-) = 3/5$$

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

2. Drop "with"

3. Likelihoods from training:

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

$$P(\text{"predictable"}|-) = \frac{1+1}{14+20} \quad P(\text{"predictable"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"no"}|-) = \frac{1+1}{14+20} \quad P(\text{"no"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"fun"}|-) = \frac{0+1}{14+20} \quad P(\text{"fun"}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

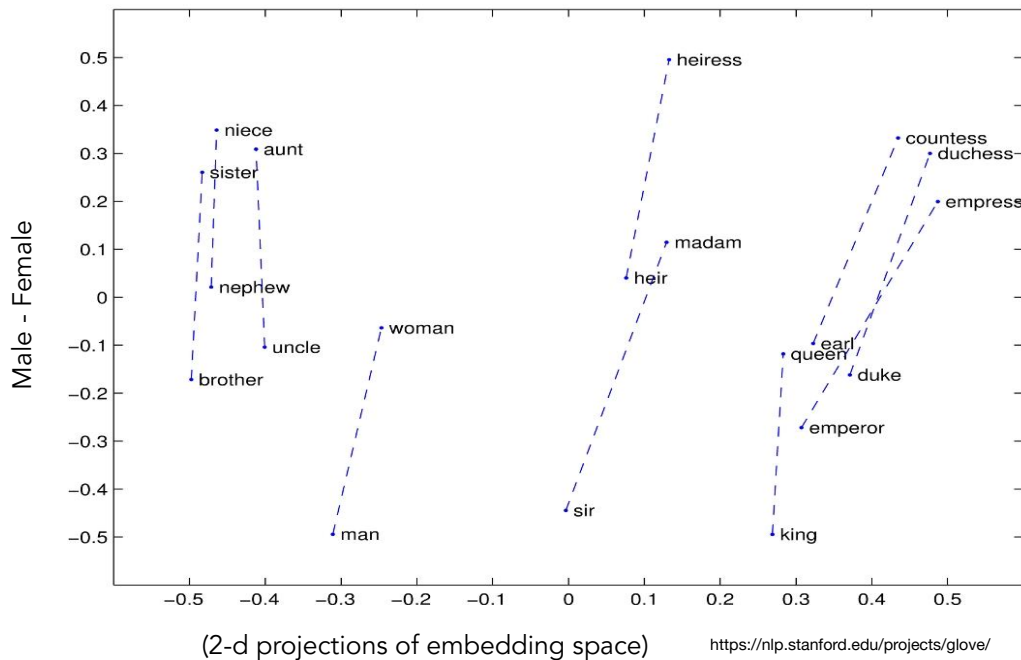
$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

In Summary

- NLP = research field at the intersection of Computer Science and Linguistics
- NLP = Many industrial applications, from Machine Translation to chatbots like ChatGPT or Information Extraction
- Token = essential unit of NLP: words, subwords, or characters
- Can count token occurrences in texts ("bag of words") to classify them

Next class: distributional semantics



A complex network of glowing blue lines and dots, resembling a molecular or digital structure, covers the right side of the slide. On the left, there is a faint, circular graphic with concentric rings and a stylized 'ai' logo in the center.

aivancity

PARIS-CACHAN

**advancing education
in artificial intelligence**

Is this spam?

Dear Paul

As per our discussions, I hereby confirm your teaching of the following courses for the 2526 academic year:

1. Natural Language Processing: PGE4 INI for S1 (24 hours)
2. Natural Language Processing: PGE4 ALT for S2 (24 hours)

Best regards,
Anuradha

Dear Dr. Lerner,

I'm [REDACTED], a representative at [REDACTED].

We provide three styles of podcast services, designed to enhance the visibility of your work:

These services come with associated production costs, starting at £1,584 for a single podcast or £4,514 for a podcast series, ideal for storytelling across three or more episodes.

Exam

- The exam will last 2 hours
- Written exam, no documents authorized
- 6/20 points on class questions ("What is X")
- 11/20 points on diverse problems
 - similar to the practical works but by hand
 - or code completion/analysis
 - or disguised class questions ("Alice wants something, how can she do it", "Bob did X, what did he do wrong")
- 3/20 points on open question