NLP and LLMs (CS40008.01)
Lecture 01 – Introduction to NLP

Baojian Zhou

NLP and LLMs (CS40008.01)

School of Data Science, Fudan University

03/05/2026

About me

Email: bjzhou@fudan.edu.cn

Website: https://baojian.github.io/

Location:

- South-401, Computing Center
- Office hour: Wed. 10:00am-11:30am

Research interests:

- Machine learning on graphs, optimization, text mining (e.g., using word embeddings), diffusion models, and in-context-learning on LLMs

群聊: 群聊

该二维码7天内(9月16日前)有效，重新进入将更新

Outline

- **Course introduction**
- **Basics for Python, nltk, spacy**
- **Tokenization**
- **Minimum edit distance**

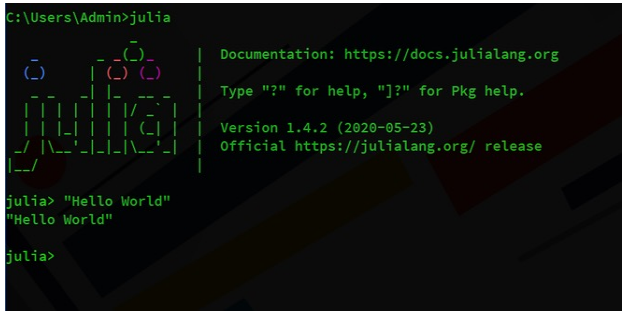
What is natural language?

- **A structured system of communication used by humans**
  - 今天天气真好!
  - The weather is so nice today!
  - 今日は天気がいいです!
  - Le temps est vraiment beau aujourd'hui!

- **Formal Language (e.g., programming languages)**

```c
#include <stdio.h>
int main(void) {
        printf("Hello, world!\n");
}
```

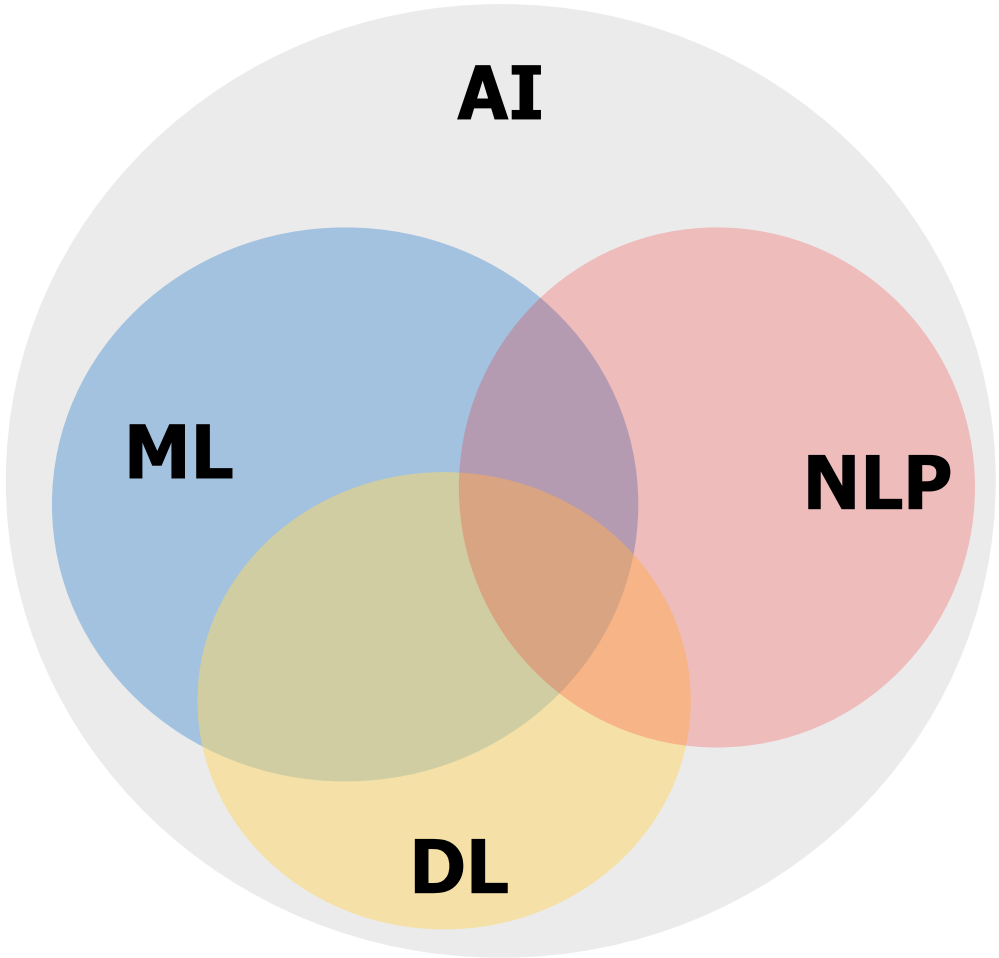
```
>>> print("Hello World!")
Hello World!
>>>
```



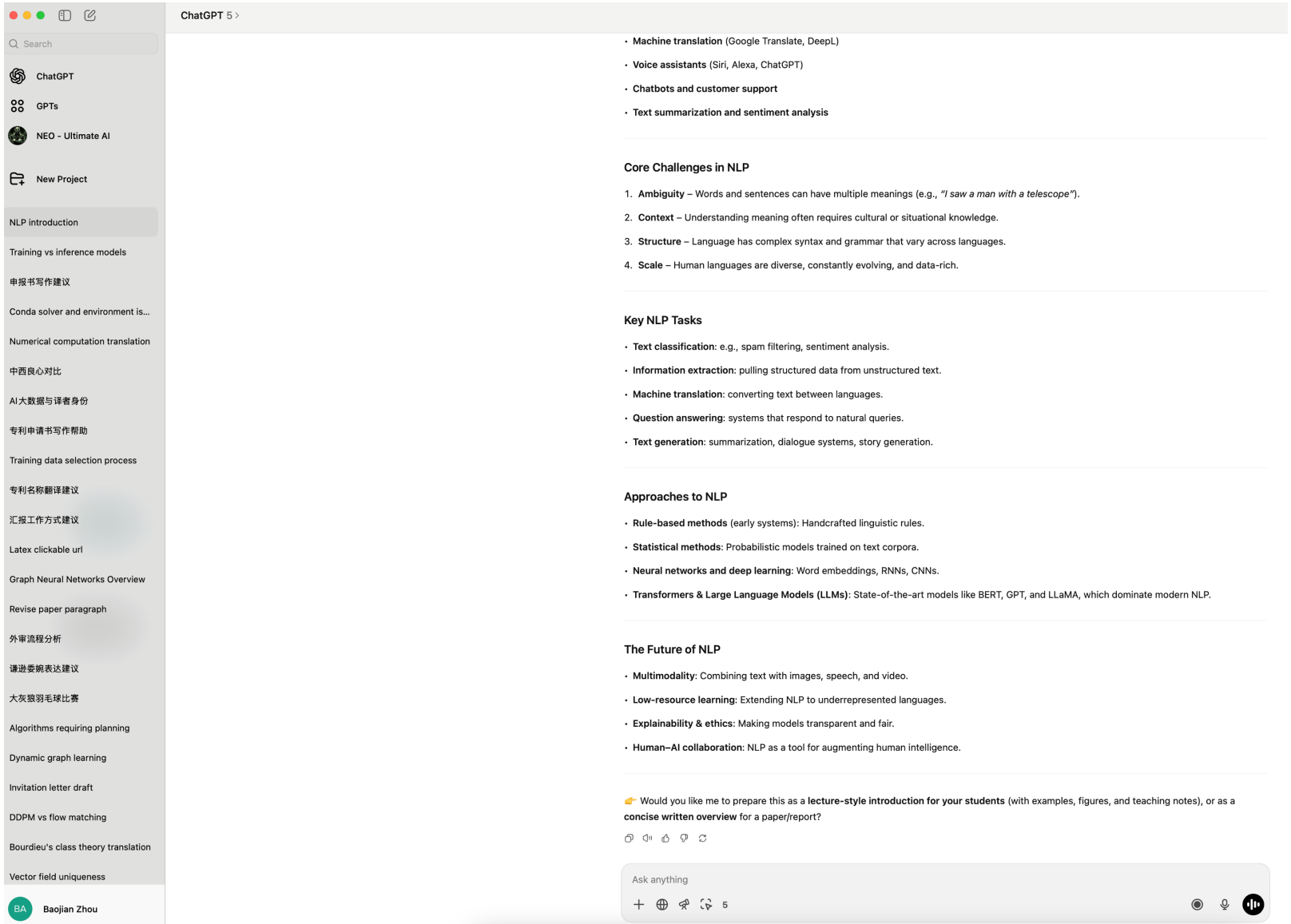
Natural Language Processing (NLP)

- **NLP is focused on enabling computers to understand, interpret, and generate human language in a way that is both meaningful and useful.**

- **Teach computers how to understand and generate human languages**
  - Natural Language Understanding
  - Natural Language Generation
- **Other names**
  - Computational Linguistics
  - Natural Language Engineering
  - Human Language Technology



Today's NLP



- [ChatGPT]
- https://www.deepseek.com/
- https://www.kimi.com/
- https://chat.qwen.ai/
- https://gemini.google.com/app
- https://www.together.ai/

NLP history (1947-1969)

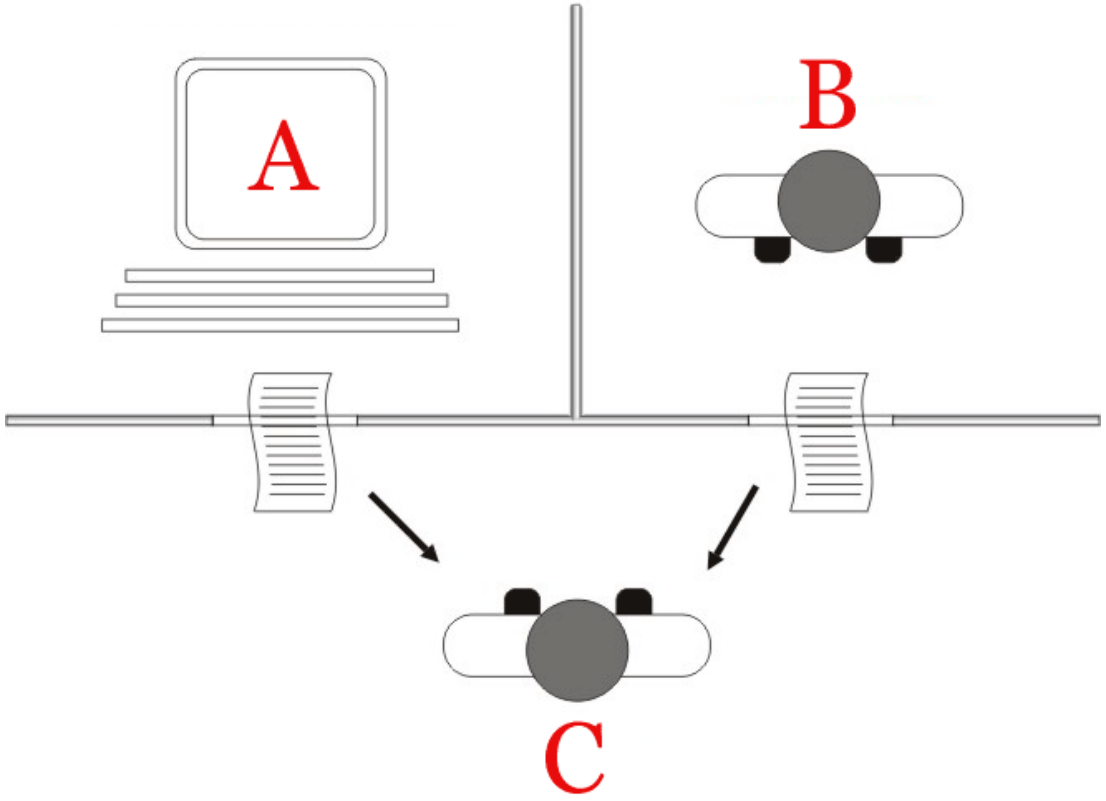
- Warren Weaver wrote to Wiener in 1947
  - *One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode'.*
- Alan Turing wrote "Computing Machinery and Intelligence" in 1950 (proposed the Turing test)
  - *I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think."*

Warren Weaver
1894-1978

Alan Turing,
1912-1954

NLP history (1947-1969)

the Turing test



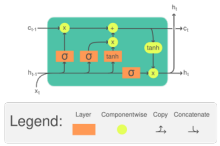
Hard part: The machine A needs to understand the person C and generate human-like languages.

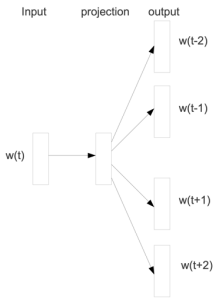
Example of Q&A:

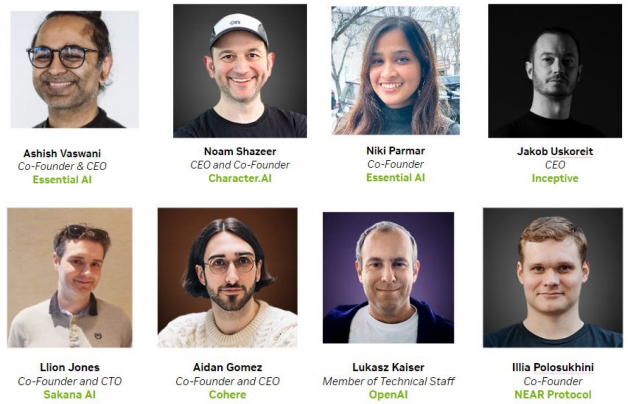
- **C: Please write me a sonnet on the subject of the Forth Bridge.**
- **A/B: Count me out on this one. I never could write poetry.**
- **C: Add 34,957 to 70,764.**
- **A/B: (Pause about 30 seconds and then give as answer) 105621.**
- **C: Do you play chess?**
- **A/B: Yes.**

- Does GPT-4 pass the Turing test?
- Large Language Models Pass the Turing Test

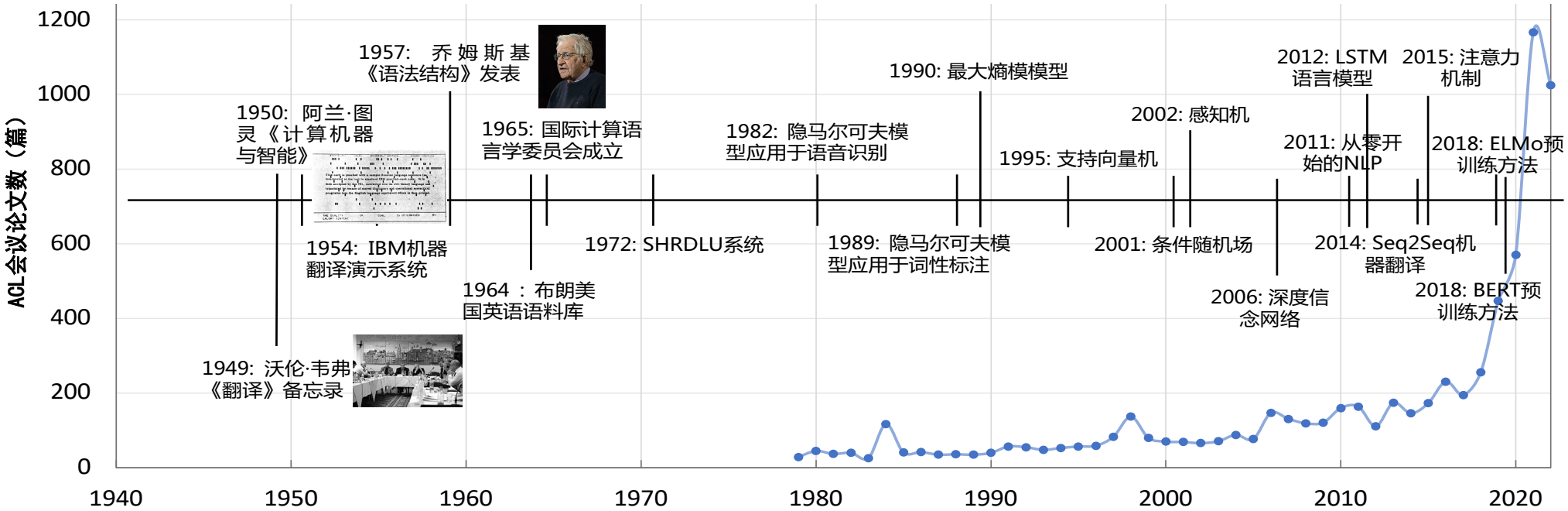
NLP history (1970-2017)

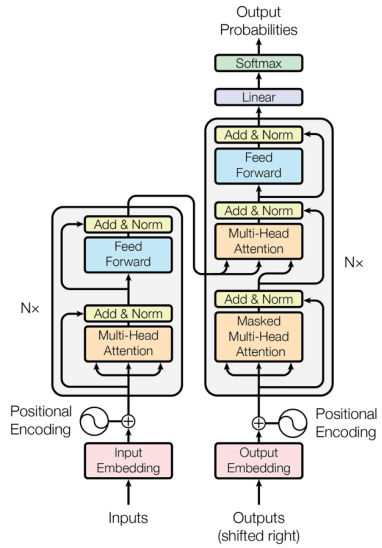




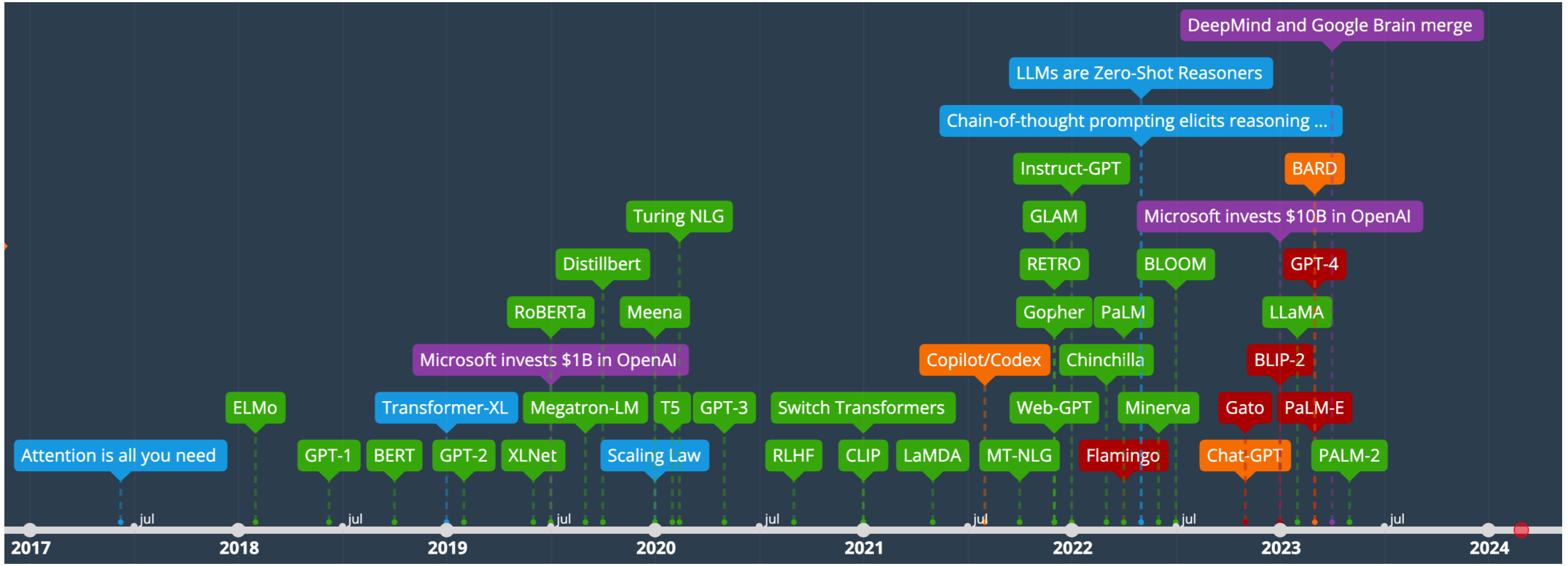


Vladimir Vapnik, SVM, 1995 **Sepp Hochreiter, LTSM, 1996** **Tomas Mikolov, word2vec, 2013** **Attention Is All You Need, 2017**





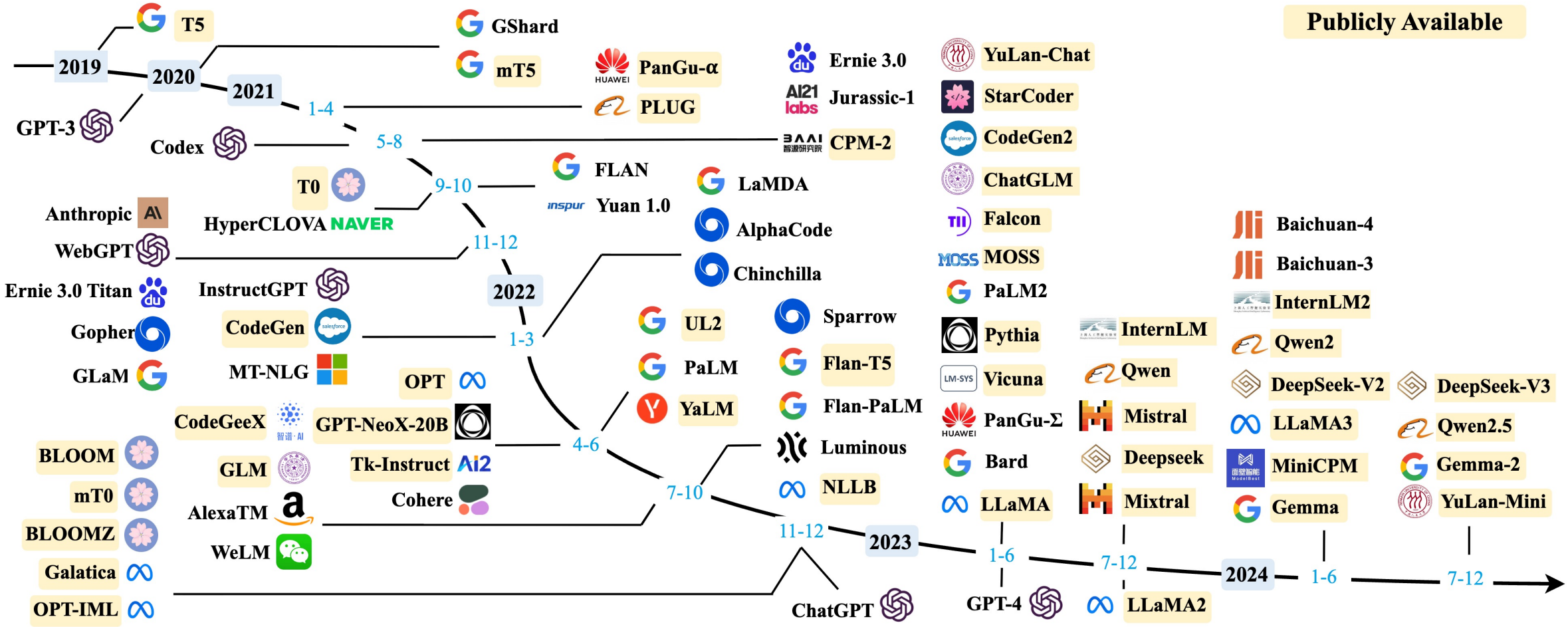
NLP history (2018-Now) - LLMs



- Bert, ELMo
- GPT-1,2,3
- ChatGPT, GPT-4: https://chat.openai.com/
- Sora: https://openai.com/sora

- By Justin Milner, https://time.graphics/line/815425
- https://ai.v-gar.de/ml/transformer/timeline/

NLP history (2018-Now) - LLMs



A Survey of Large Language Models

What is next?

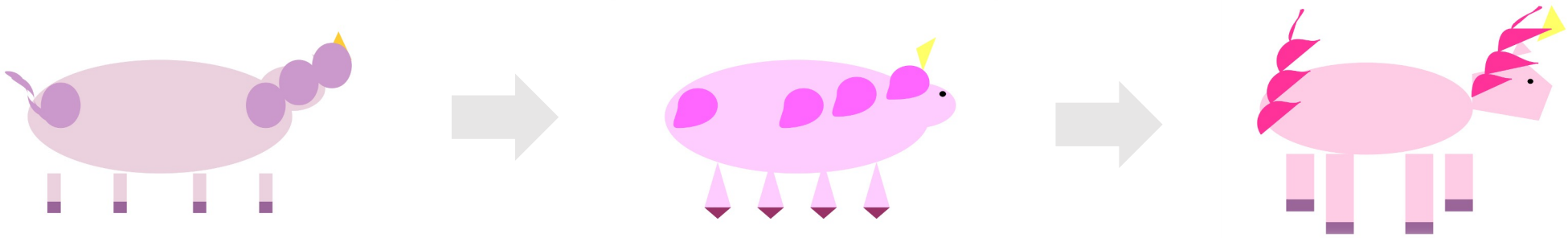
- Sparks of Artificial General Intelligence (AGI)

Sparks of Artificial General Intelligence:
Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

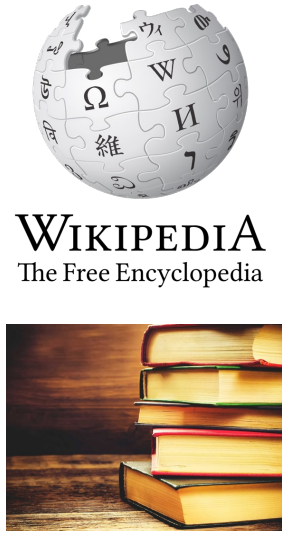
- Prompt: "Draw a unicorn in TikZ" (Query GPT-4 at roughly equal time intervals while the system was being refined)

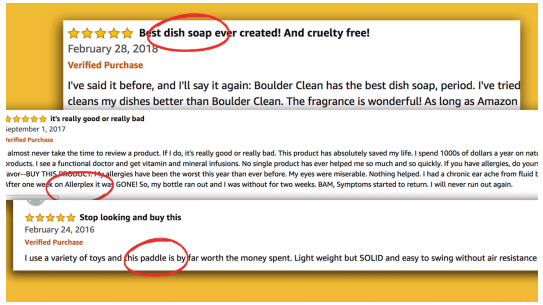


Text data – written text

Sources

- Blogs
- Microblogs
- Forums
- Reviews
- Books





Topics

- People
- Events
- Products

Task: find out the pre-training datasets of the following models

- DeepSeek-V3, Qwen3, Llama 3, Gemini 2.5

Tasks - sentiment analysis

- **Attributes**
  - Zoom
  - Affordability
  - Size and weight
  - Flash
  - Ease of use

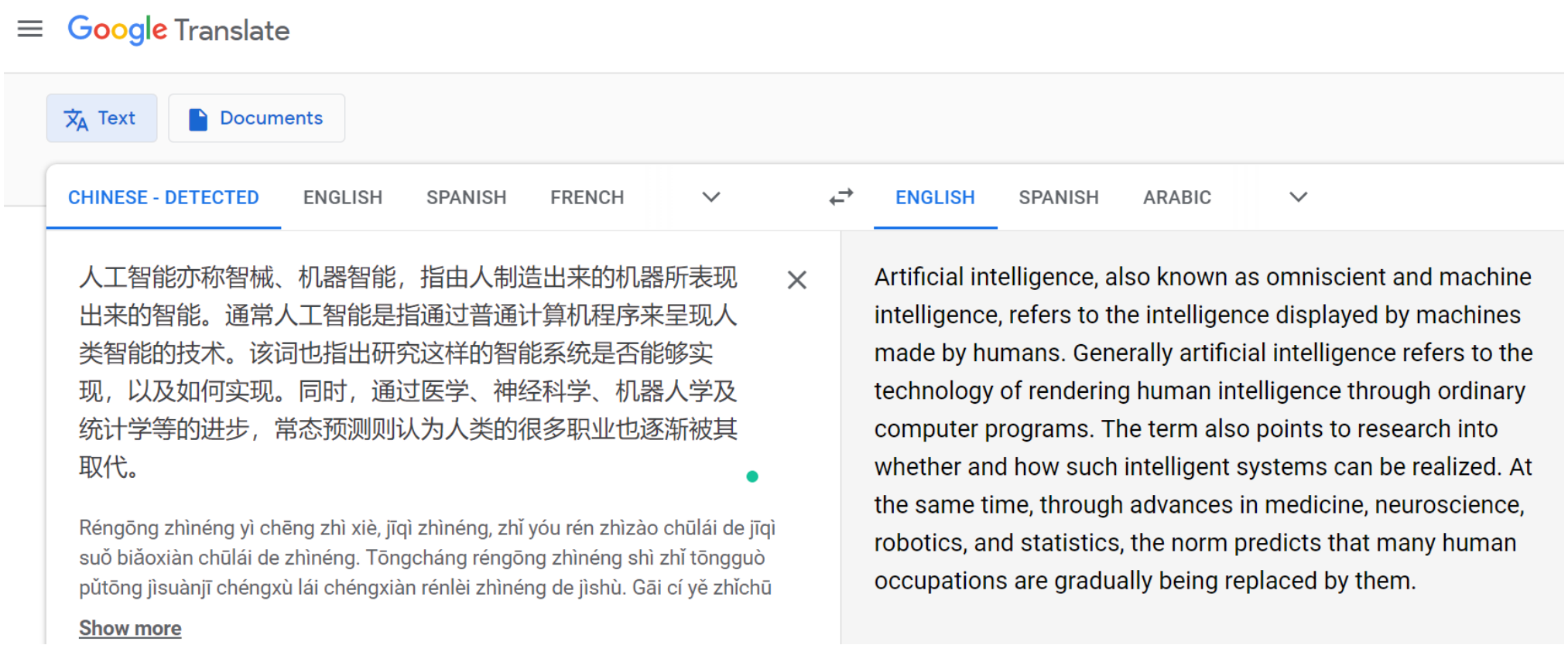
✓ • Nice and compact to carry!
  - **Positive**

✓ • Since the camera is small and light, I won't need to carry around those heavy, bulky professional cameras either!
  - **Positive**

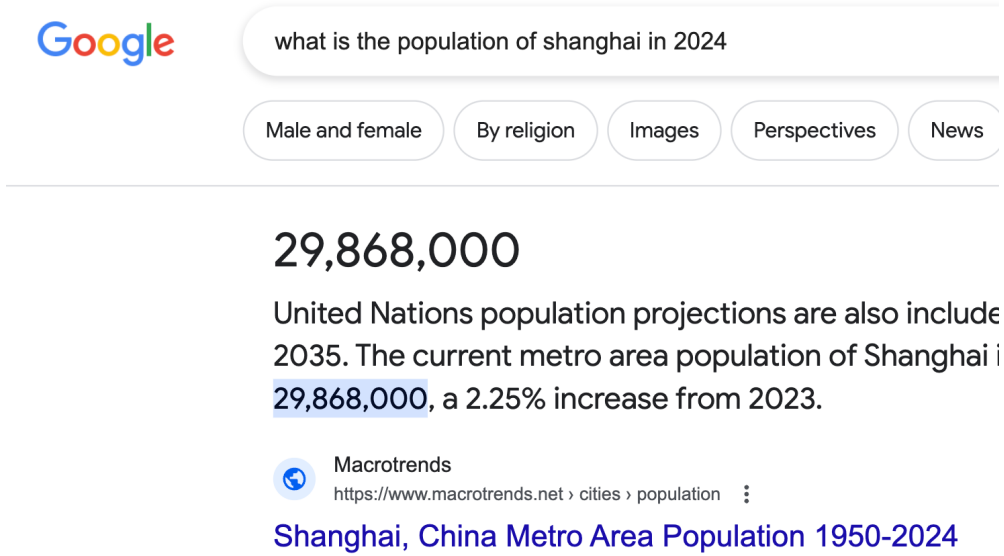
✗ • The camera feels flimsy, is plastic and very light in weight you have to be very delicate in the handling of this camera.
  - **Negative**

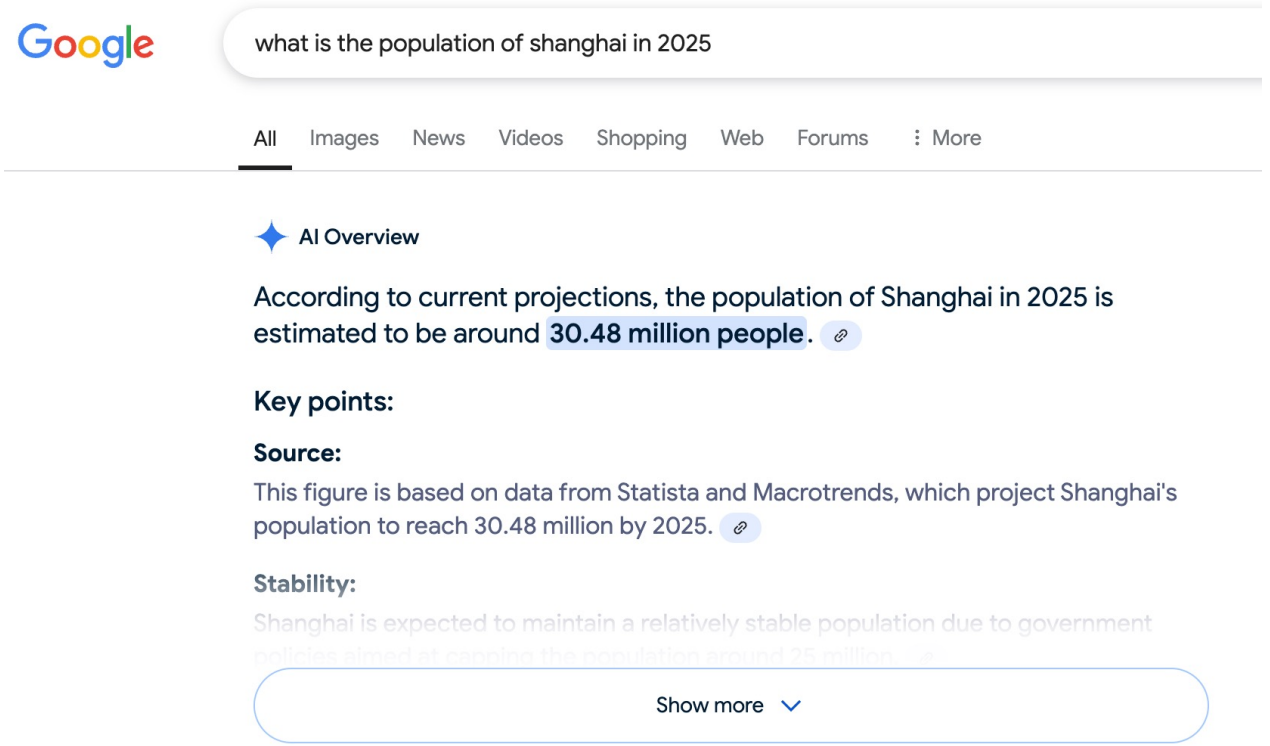
Tasks - machine translation

Chinese → English



Tasks - question and answering





- Provide answers to question directly
- Advance version of information retrieval

Tasks – article generation

A: Blog

- **Title**: Feeling unproductive? Maybe you should stop overthinking.
- **Content**: In order to get something done, maybe we need to think less. Seems counter-intuitive, but I believe sometimes our thoughts can get in the way of the creative process. We can work better at times when we "tune out" the external world and focus on what's in front of us. I've been thinking about this lately, so I thought it would be good to write an article about it…

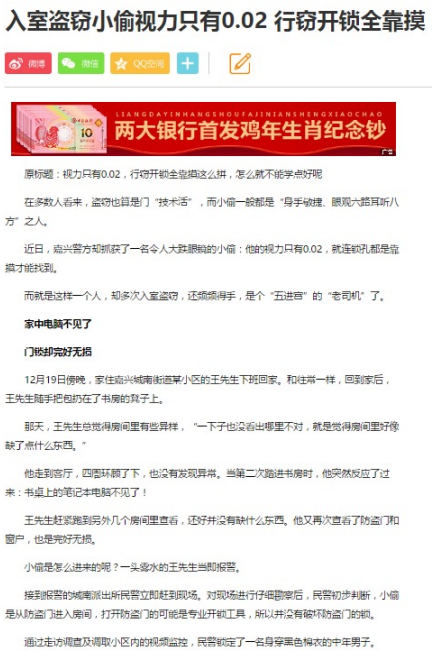
B: News article

- **Title**: United Methodists Agree to Historic Split
- **Content**: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan…

1. **A: Human Written B: Human Written**
2. **A: Machine Written B: Human Written**
3. **A: Human Written B: Machine Written**
4. **A: Machine Written B: Machine Written**

They are all generated by GPT-3!

Tasks – text summarization



入室盗窃小偷视力只有0.02 行窃开锁全靠摸

原标题：视力只有0.02，行窃开锁全靠摸这么拼，怎么就不能学点好呢

在多数人看来，盗窃也算是门“技术活”，而小偷一般都是“身手敏捷、眼观六路耳听八方”之人。

近日，嘉兴警方却抓获了一名令人大跌眼镜的小偷：他的视力只有0.02，就连锁孔都是靠摸才能找到。

而就是这样一个人，却多次入室盗窃，还频频得手，是个“五进宫”的“老司机”了。

家中电脑不见了

门锁却完好无损

12月19日傍晚，家住嘉兴城南街道某小区的王先生下班回家。和往常一样，回到家后，王先生随手把包扔在了书房的凳子上。

那天，王先生总觉得房间里有些异样，“一下子也没看出哪里不对，就是觉得房间里好像缺了点什么东西。”

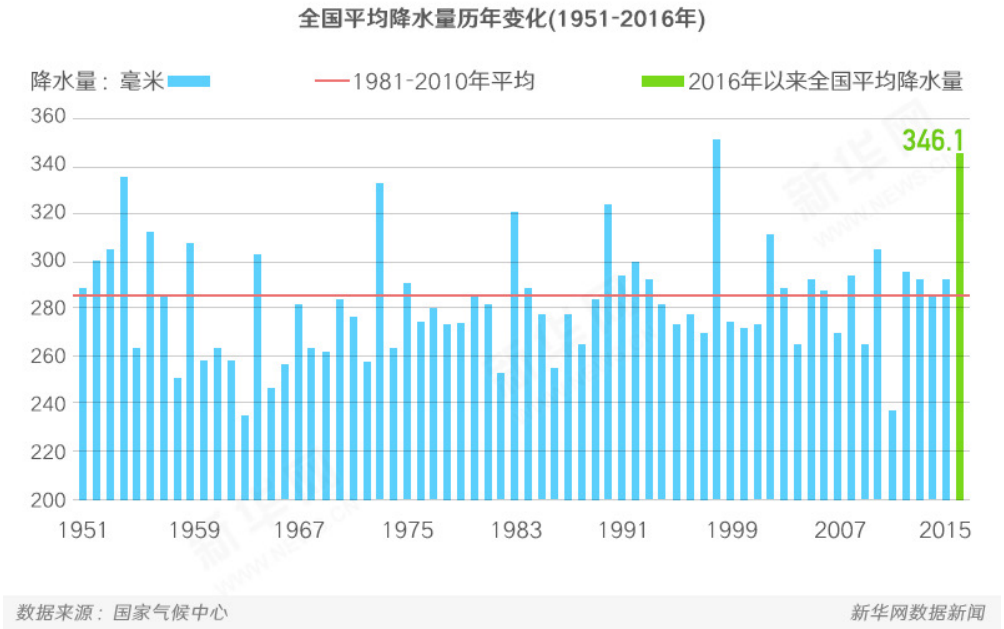
他走到客厅，四面环顾了下，也没有发现异常。当第二次踏进书房时，他突然反应了过来：书桌上的笔记本电脑不见了！

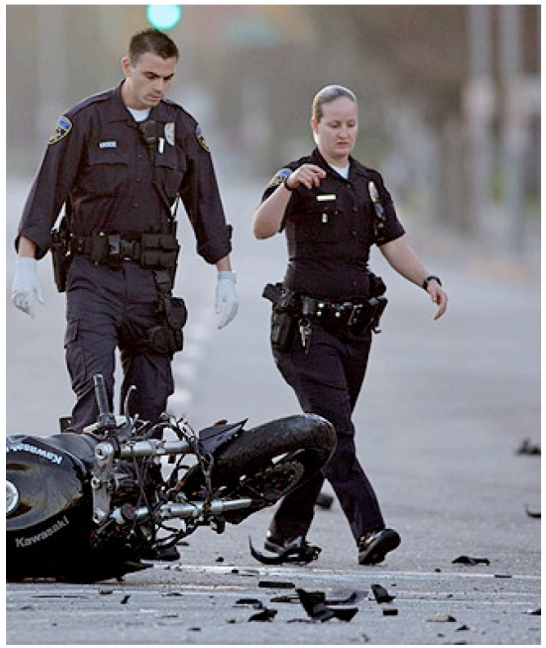
王先生赶紧跑到另外几个房间里查看，还好并没有缺什么东西。他又再次查看了防盗门和窗户，也是完好无损。

小偷是怎么进来的呢？一头雾水的王先生当即报警。

接到报警的城南派出所民警立即赶到现场。对现场进行仔细勘察后，民警初步判断，小偷是从防盗门进入房间，打开防盗门的可能是专业开锁工具，所以并没有破坏防盗门的锁。

通过走访调查及调取小区内的视频监控，民警锁定了一名身穿黑色棉衣的中年男子。





News Rewriting

从1981年到2010年，全国平均降水量为280毫米 …

Summarization from images

Tasks – build Chatbot

A: Men are all alike

B: In what way?

A: They're always bugging us about something or other.

B: Can you think of a specific example?

A: Well, my boyfriend made me come here.

B: Your boyfriend made you come here?

A: He says I'm depressed much of the time.

B: I'm sorry to hear you are depressed.

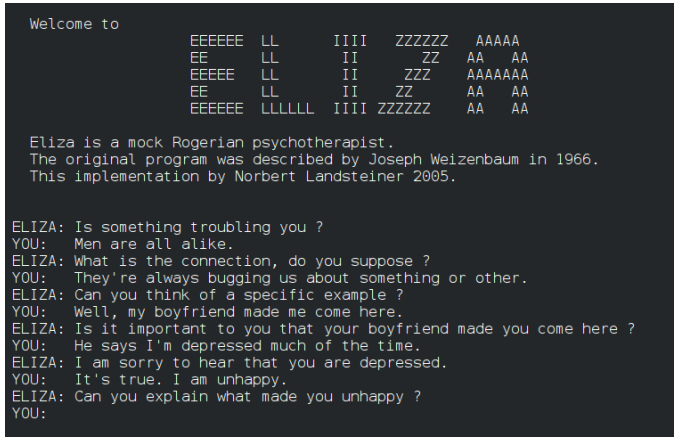
A: It's true. I am unhappy.

B: Do you think coming here will help you not to be unhappy?

A: I need some help, that much seems certain....

1. **A: Human** **B: Human**
2. **A: Machine** **B: Human**
3. **A: Human** **B: Machine**
4. **A: Machine** **B: Machine**

A: Human B: Machine



[PDF] **ELIZA—a computer program** for the **study** of **natural language communication between man** and **machine** [PDF] acm.org

J Weizenbaum - **Communications** of the ACM, 1966 - dl.acm.org

… ELIZA is a **program** operating within the MAC time-sharing system at MIT which makes certain kinds of **natural language** conversation **between man** and **computer** possible. Input sen…

☆ Save 99 Cite Cited by 5579 Related articles All 30 versions

Tasks – text to image

- Creating image from text IO (OpenAI, Sora)

Vibrant coral reef teeming with colorful fish and sea creatures

A snowy mountain village with cozy cabins and a northern lights display, high detail and photorealistic dslr, 50mm f/1.2

Close-up portrait shot of a woman in autumn, extreme detail, shallow depth of field

Tasks – text to video

- **Creating video from text:** a woman wearing purple overalls and cowboy boots taking a pleasant stroll in Johannesburg South Africa during a beautiful sunset

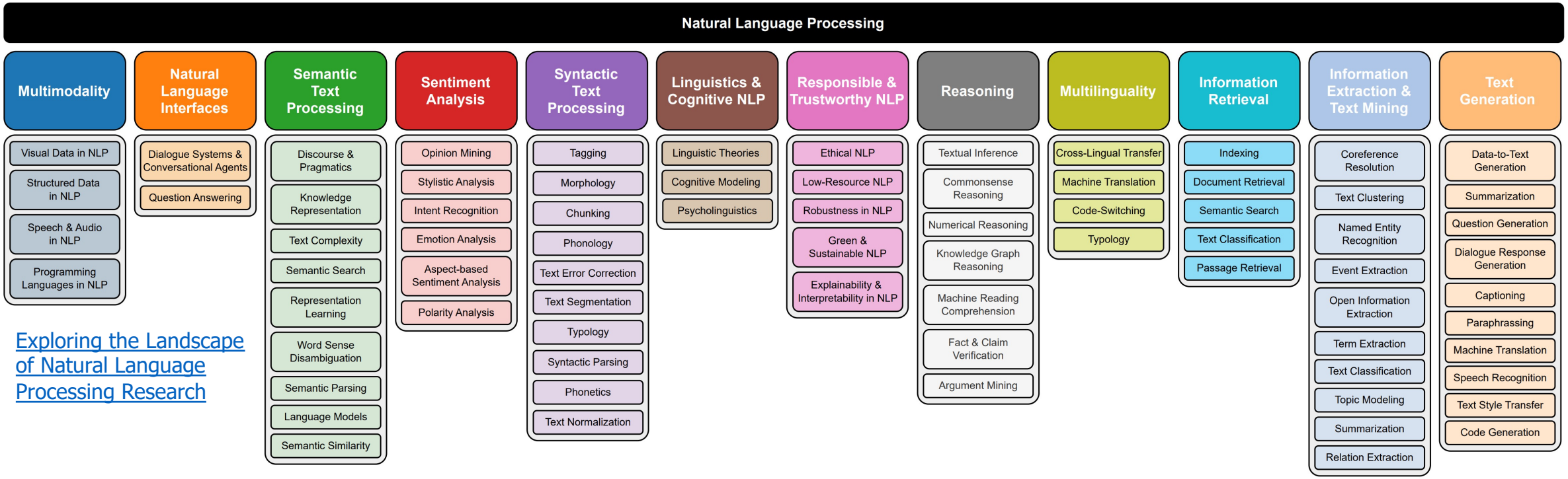
OpenAI, Sora

Tasks – text to video

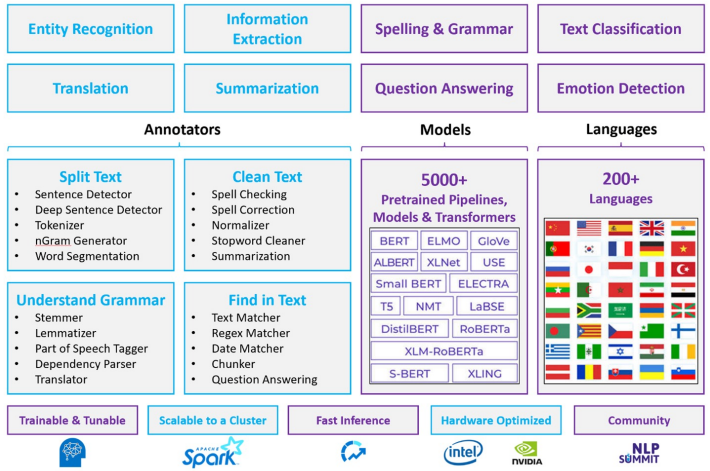
- **Creating video from text**: an old man wearing blue jeans and a white T-shirt taking a pleasant stroll in Antarctica during a winter storm

OpenAI, Sora

Many other NLP tasks/products



[Exploring the Landscape of Natural Language Processing Research]



Predicted before LLMs era

If I were starting a company today, it would use AI to teach computers how to read.

NLP is difficult!

- **Ambiguity makes NLP hard!**
  - **A man saw a boy *with a telescope*.** — Who had the telescope?
  - **He has *quit* smoking.** — It means that he smoked before.
  - What does the ***Mighty Dragon*** mean?

Ambiguities in English/Chinese

- Teacher strikes idle kids.
- The chickens are too hot to eat.
- The old men and women left the room.
- He left her in tears.
- Hospitals are sued by 6 foot doctors.

- 冬天，能穿多少穿多少；夏天，能穿多少穿多少；
- 单身的来由：原来是喜欢一个人，现在是喜欢一个人；
- 女致电男友：地铁站见。如果你到了我还没到，你就等着吧。如果我到了你还没到，你就等着吧！

Hard to inference

- **Example**
  - A dime is better than a nickel
  - A nickel is better than a penny
  - Therefore, a dime is better than a penny
- **Adversarial Example**
  - A penny is better than nothing
  - Nothing is better than world peace.
  - Therefore, a penny is better than world peace ???

Other difficulties

Non-standard English

Great job @justinbieber! Were SOO PROUD of what youve accomplished! U taught us 2 #neversaynever & you yourself should never give up either♥

Segmentation issues

San Francisco
Los Angeles

the New York-New Haven Railroad
the New York-New Haven Railroad

Idioms

dark horse
lose face
break a leg
bite the bullet

Neologisms

unfriend
Retweet
bromance
鸡娃

World knowledge

Mary and Sue are sisters.
Mary and Sue are mothers.

Tricky entity names

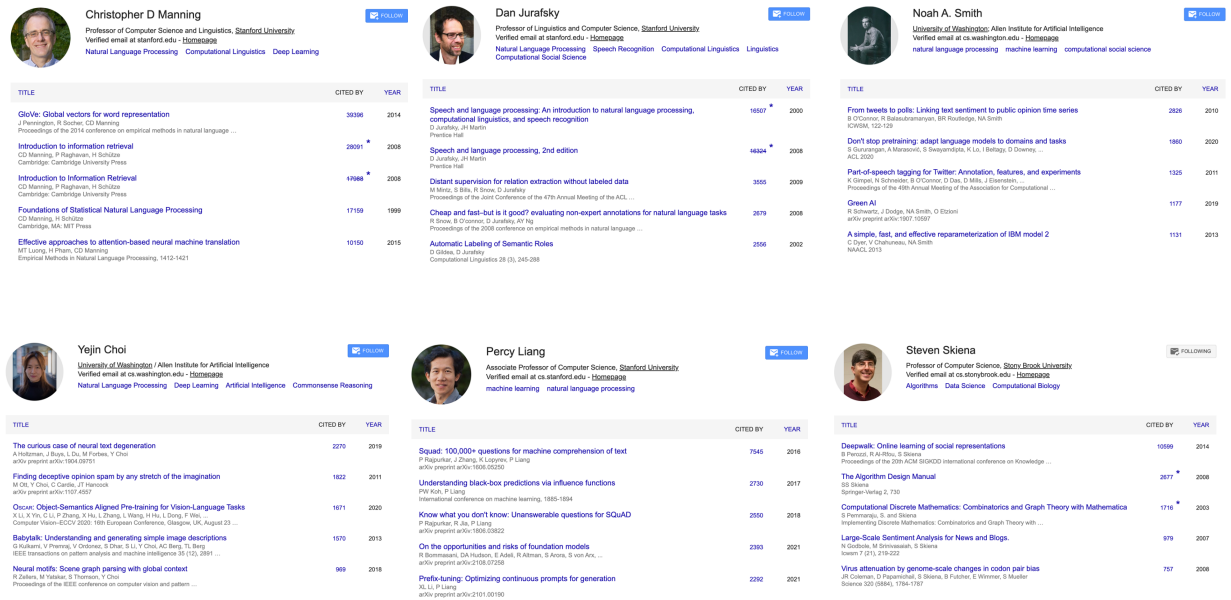
Where is ***A Bug's Life*** playing …

Let It Be was recorded …

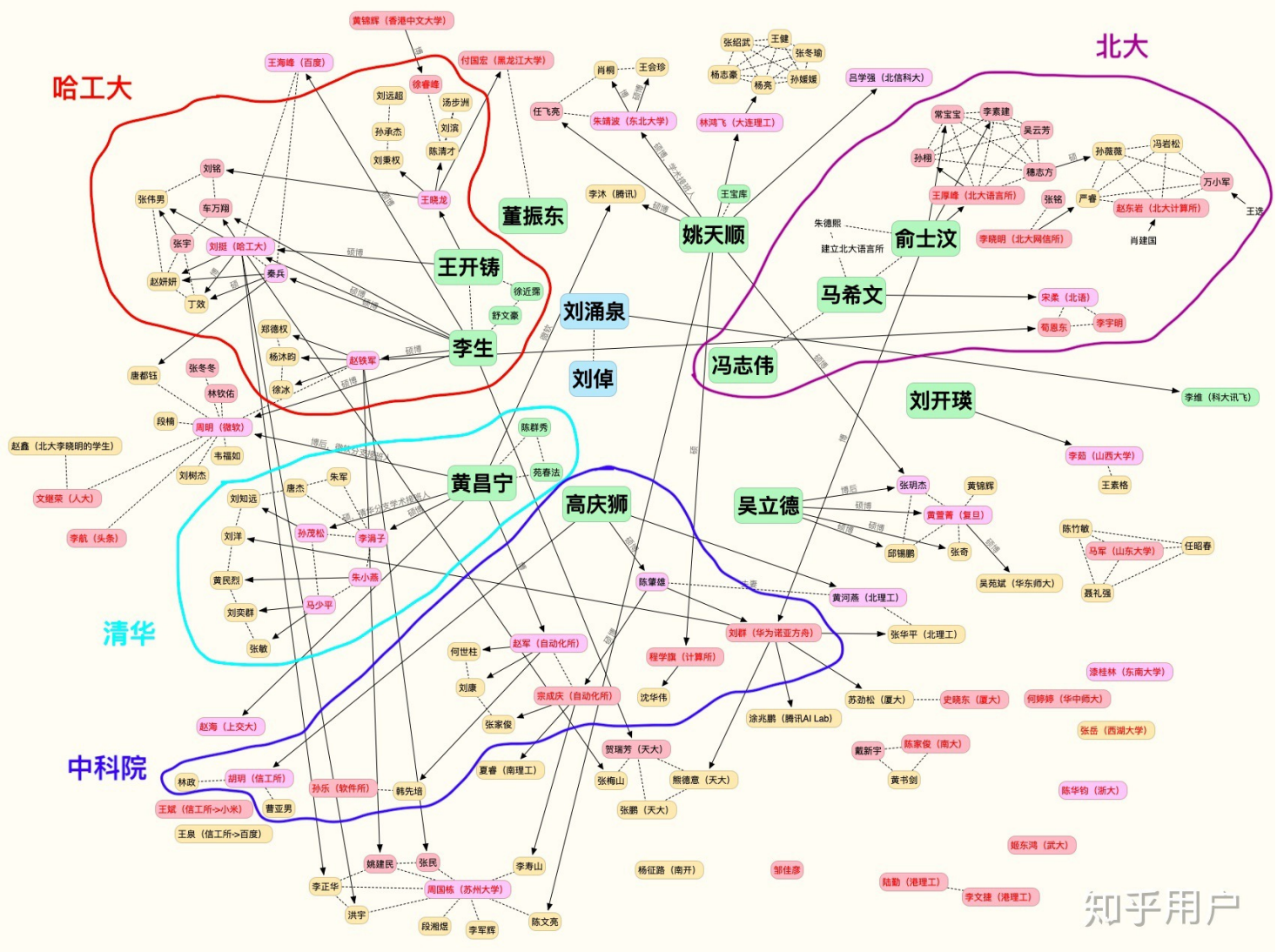
NLP conferences and people

- **Natural Language Processing**
  - ACL, EMNLP, COLING, NAACL, EACL (https://aclanthology.org/)
- **Machine learning**
  - ICML, NIPS, ICLR
- **Information Retrieval**
  - SIGIR, WWW, CIKM, WSDM
- **Data Mining**
  - KDD

People in NLP



People in NLP (China)



https://www.zhihu.com/question/24366306

CSRankings: Computer Science Rankings

CSRankings is a metrics-based ranking of top computer science institutions around the world. **Click on a triangle** (▶) to expand areas or institutions. **Click on a name** to g faculty member's home page. **Click on a chart icon** (the after a name or institution) to see the distribution of their publication areas as a bar chart **Click on a Goog Scholar icon** (ⓢ) to see publications, and **click on the DBLP logo** (🄳) to go to a DBLP entry.
Applying to grad school? Read this first.
Rank institutions in the world by publications from 2015 to 2022

All Areas [off | on]

AI [off | on]
- ▶ Artificial intelligence
- ▶ Computer vision
- ▶ Machine learning & data mining
- ▶ Natural language processing
- ▶ The Web & information retrieval

Systems [off | on]
- ▶ Computer architecture
- ▶ Computer networks
- ▶ Computer security
- ▶ Databases
- ▶ Design automation
- ▶ Embedded & real-time systems
- ▶ High-performance computing
- ▶ Mobile computing
- ▶ Measurement & perf. analysis
- ▶ Operating systems
- ▶ Programming languages
- ▶ Software engineering

Theory [off | on]
- ▶ Algorithms & complexity
- ▶ Cryptography
- ▶ Logic & verification

4	▶ University of Washington	61.4	9
5	▶ Harbin Institute of Technology	54.3	30
6	▶ Tsinghua University	52.5	24
7	▶ Chinese Academy of Sciences	43.2	15
8	▶ Cornell University	41.2	12
9	▶ University of Pennsylvania	37.6	8
10	▼ Fudan University	37.2	14

Faculty	*# Pubs*	*Adj. #*
Xuanjing Huang NLP,AI	56	12.5
Xipeng Qiu NLP	38	8.8
Qi Zhang 0001 AI,NLP	20	4.0
Zhongyu Wei NLP	17	3.3
Xiaoqing Zheng AI	8	2.6
Yaqian Zhou NLP	7	1.6
Yanghua Xiao AI	6	1.4
Xiangyang Xue VISION	2	0.4
Wei Wang 0009 AI	1	0.3
Zengfeng Huang ML	1	0.1
Yu-Gang Jiang AI,VISION	1	0.3
Zhipeng Xie AI	1	0.5
Yanchun Zhang AI	1	0.1
Shuigeng Zhou AI,VISION	1	0.2

邱锡鹏老师的回答：
https://www.zhihu.com/question/24366306/answer/123787923

知識工場

What we will cover? (tentative)

- **(Basics) Text pre-processing and tokenization**
- **(Basics) Language models (N-grams)**
- **(Basics) Word embeddings (word2vec)**
- **(Basics) Neural networks (NNs) for LM**
- **(Basics) RNNs for Sequence labeling**
- **(Basics) Self-attention mechanism**
- **(Basics) Transformer architecture**
- **(LLMs) Pretraining and fine-tuning**
- **(LLMs) Evaluation and benchmarking**
- **(LLMs) In-context-learning**
- **(LLMs) Reasoning and Agents**
- **(LLMs) Diffusion Language Models**
- **(Applications) Machine translation**
- **(Applications) Syntactic analysis**

Skills needed

- **Linear algebra (vectors, matrices)**
- **Basic Statistics / Machine Learning algorithms**
- **Basic Python programming skills (numpy, pytorch)**
- **Communication skills (team project)**

Coursework

- **Participation (5%)**
- **Quizzes (10%)**
- **Assignments (45-50%)**
- **Final Project (35-40%)**
- **Office hours**
  - **Lecturer:** Baojian Zhou, Wed. 10:00-11:30am
  - **TA:** Binbin Huang（黄彬彬, 24210980091@m.fudan.edu.cn),

    Yuxiang Wang (王煜祥, 25210980109@m.fudan.edu.cn)

Submit to Fudan eLearning: https://elearning.fudan.edu.cn/

Integrity policy: http://xxgk.fudan.edu.cn/bd/61/c5163a48481/page.htm

Textbooks

- **Dan Jurafsky and James H. Martin, Speech and Language Processing, https://web.stanford.edu/~jurafsky/slp3/**
- Jacob Eisenstein, Natural Language Processing, 2018
- 张奇、桂韬、黄萱菁，自然语言处理导论, https://intro-nlp.github.io/, 2022

Other useful resources

- Stanford CS 224N: Deep learning for NLP, Christopher Manning
- CMU 11-711 ANLP: Advanced NLP, Graham Neubig
- UMass CS685: Advanced NLP, Mohit Iyyer
- COS 484: https://princeton-nlp.github.io/cos484/ , Danqi Chen
- Online courses: Neural Networks: Zero to Hero, Andrej Karpathy
- Stanford CS 124: From Languages to Information, Dan Jurafsky

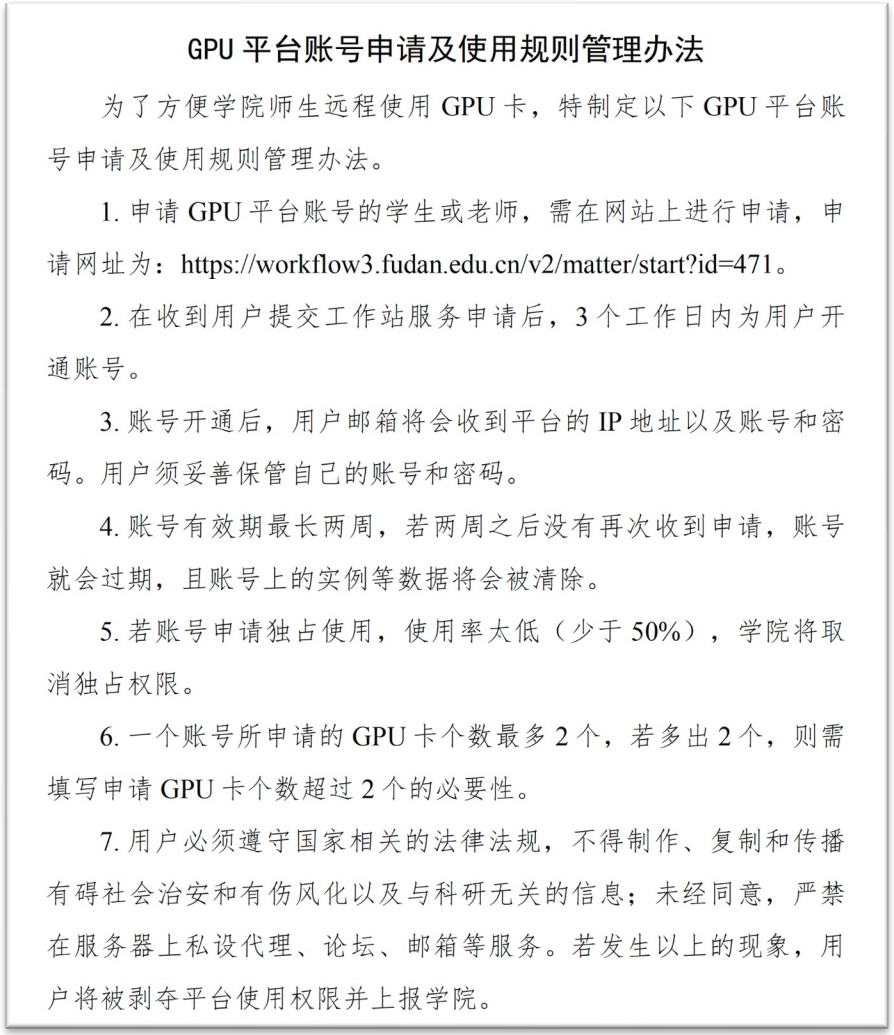
GPU resources

GPU卡申请表

申请人	XXX	学工号	XXX
类别	请选择 *	导师	*
邮箱	请输入内容 *	联系电话	请输入内容 *
工作站开始日期	2024-02-26 *	工作站结束日期	2024-02-26 *
申请使用资源 （GPU申请个数）	请输入内容 *	用途（需写具体名称，如毕业论文：XXX）	NLP课程PJ：XXX *
申请GPU卡个数超过2个的必要性	请输入内容		
是否共享	○ 可共享（共享最长可申请时间为两周） ○ 不可共享（不可共享最长可申请时间为一周） *		
签字（导师）			
审签（管理员）			

注意事项：

1. 邮箱请填写后仔细核对，该地址将收到申请结果
2. 若跑的数据集特别大（几十G）的话，网卡可能会被卡死，所以该平台只适合小量数据集
3. 请随时保存自己的数据，使用结束后删除自己的实例
4. 若使用过程中使用率少于50%，学院将减少GPU申请个数



GPU 平台账号申请及使用规则管理办法

为了方便学院师生远程使用 GPU 卡，特制定以下 GPU 平台账号申请及使用规则管理办法。

1. 申请 GPU 平台账号的学生或老师，需在网站上进行申请，申请网址为：https://workflow3.fudan.edu.cn/v2/matter/start?id=471。

2. 在收到用户提交工作站服务申请后，3 个工作日内为用户开通账号。

3. 账号开通后，用户邮箱将会收到平台的 IP 地址以及账号和密码。用户须妥善保管自己的账号和密码。

4. 账号有效期最长两周，若两周之后没有再次收到申请，账号就会过期，且账号上的实例等数据将会被清除。

5. 若账号申请独占使用，使用率太低（少于 50%），学院将取消独占权限。

6. 一个账号所申请的 GPU 卡个数最多 2 个，若多出 2 个，则需填写申请 GPU 卡个数超过 2 个的必要性。

7. 用户必须遵守国家相关的法律法规，不得制作、复制和传播有碍社会治安和有伤风化以及与科研无关的信息；未经同意，严禁在服务器上私设代理、论坛、邮箱等服务。若发生以上的现象，用户将被剥夺平台使用权限并上报学院。

Outline

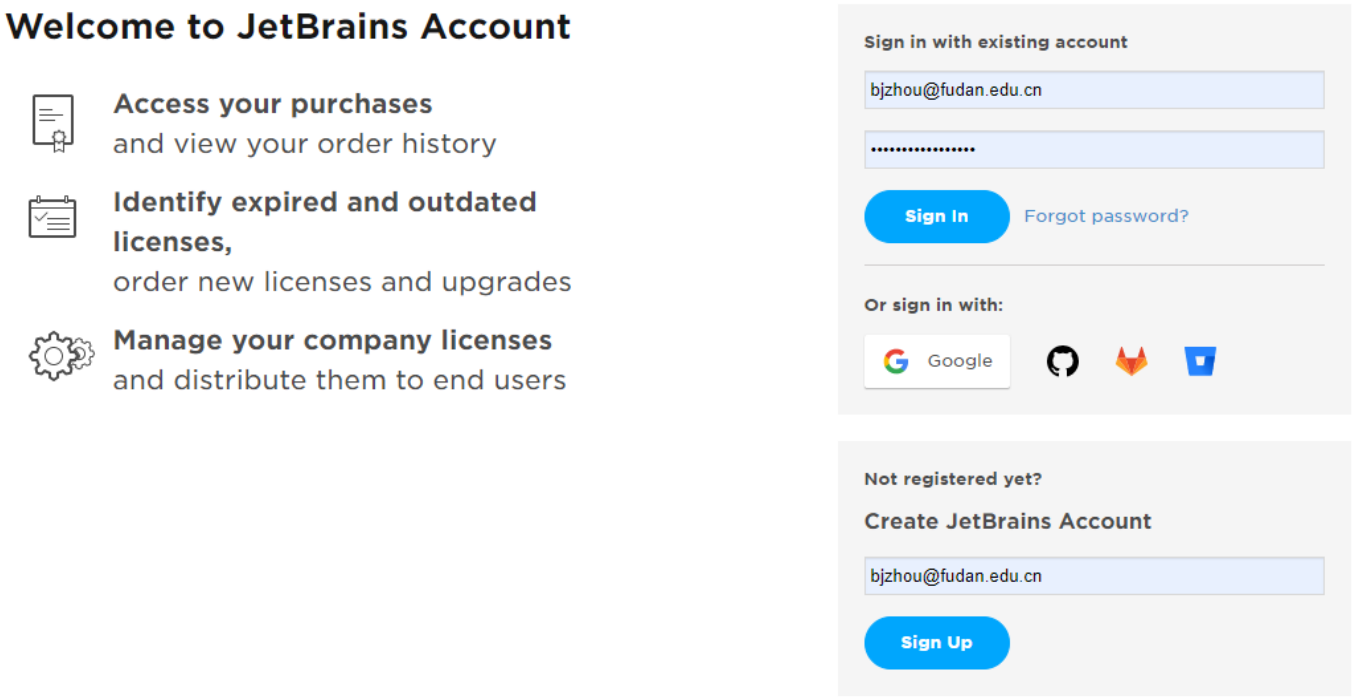
- Course introduction
- **Basics for Python, nltk, spacy**
- Tokenization
- Minimum edit distance

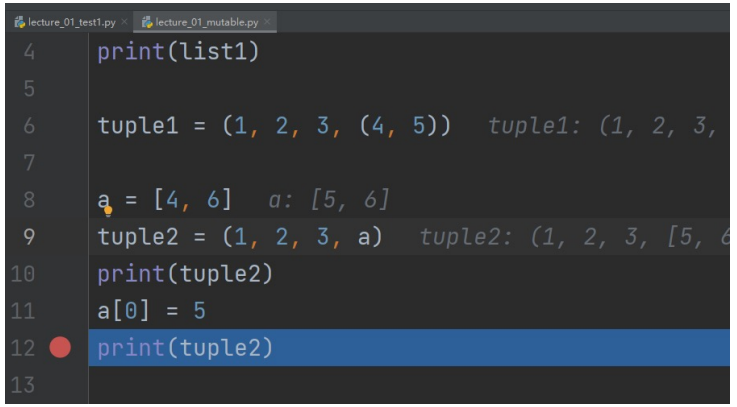
Install Python & Tools

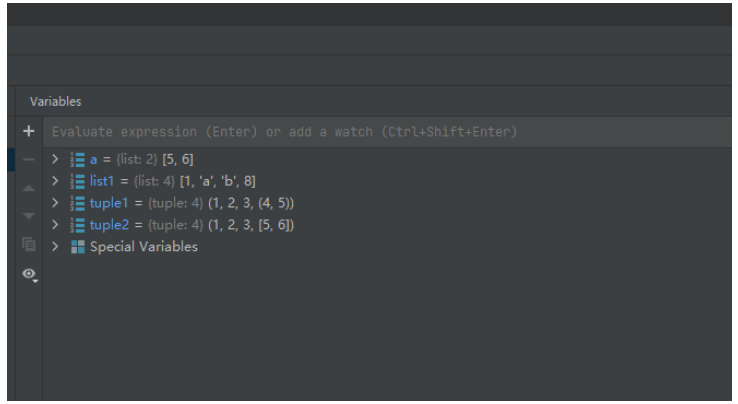
- **Python**
  - https://www.python.org/downloads/ Python3.10
- **Python IDE**
  - **PyCharm** - https://www.jetbrains.com/pycharm/
  - **Jupyter notebook** - https://jupyter.org/
  - **VS Code** - https://code.visualstudio.com/
- **Anaconda**
  - https://www.anaconda.com/ (Strongly Recommended)
- **Git**
  - https://github.com/ to manage your code

PyCharm is free for students

Use your fudan.edu.cn email account







Debug on PyCharm

Cursor: https://cursor.com/home

AI tools Cursor

Install anaconda and Python env

- Download: https://docs.anaconda.com/anaconda/install/index.html
- Activate: **source ~/anaconda3/bin activate**
- Create env: **conda create -n llm-26 python=3.12**
- Activate env: **conda activate llm-26**
- Type python, you will go into python env under **llm-26**

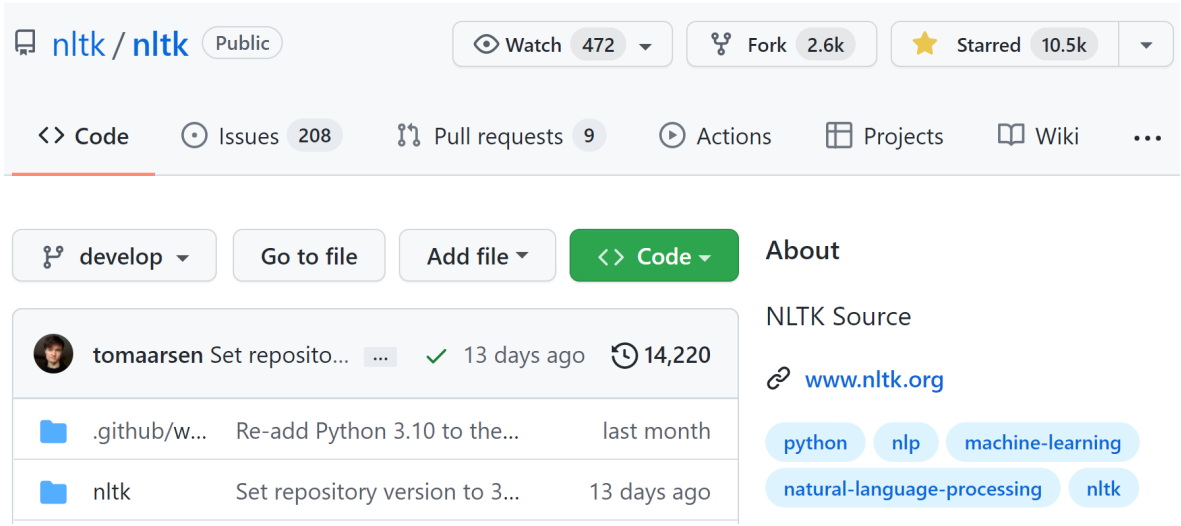
Jupyter notebook

- Install: *conda install -c conda-forge jupyterlab*
- Open your notebook:
- Download some text data: https://www.nltk.org/data.html

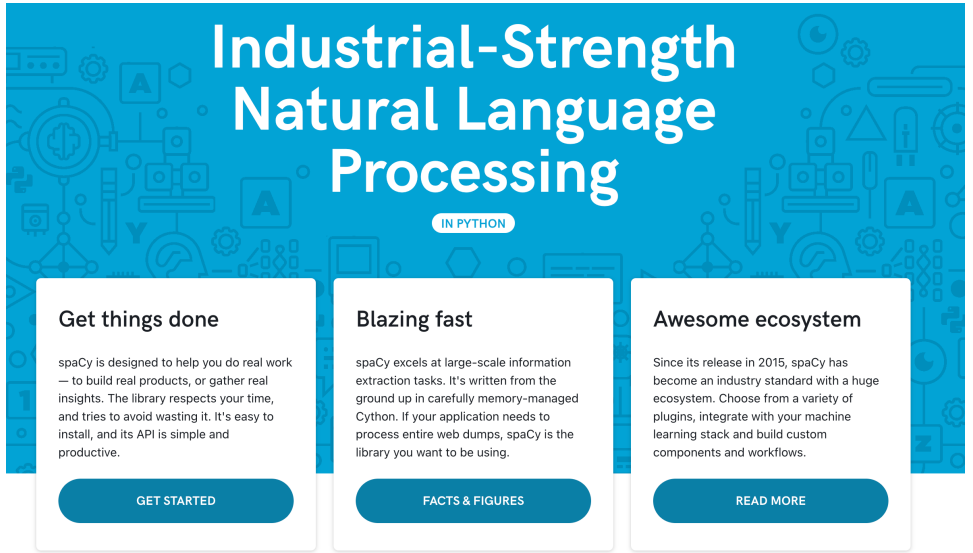
Packages need to install

- Packages （This lecture: nltk, spacy, Jupyter, matplotlib, transformers)
- Package Install: conda install xxx, pip install (not to mix them)
- Use Package： import nltk

NLTK



spaCy

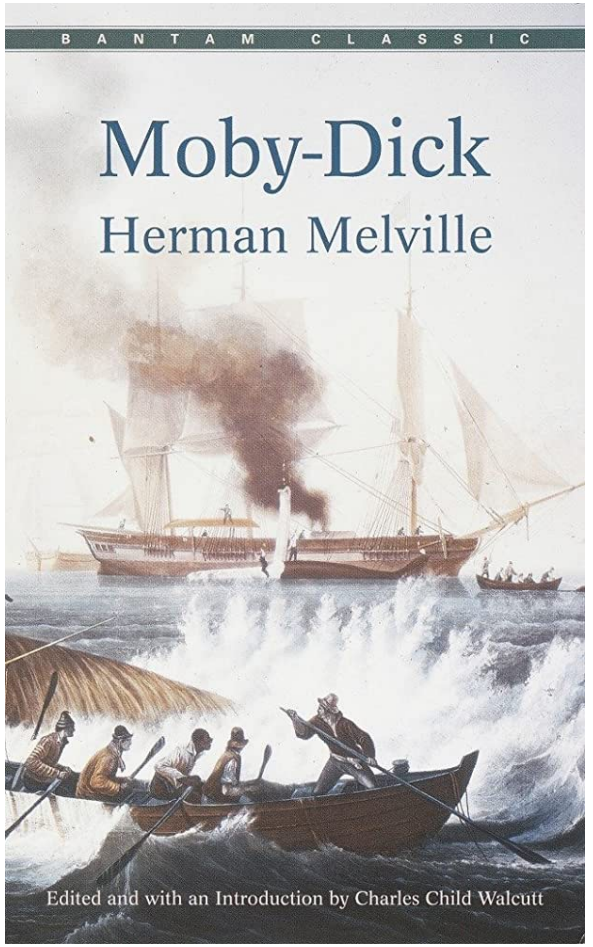


Text searching

- In what kind of contexts is the word *monstrous* typically used?

```
from nltk.corpus import gutenberg
from nltk.text import Text
corpus = gutenberg.words('melville-moby_dick.txt')
text = Text(corpus)
text.concordance("monstrous")
```

```
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```



Try our demo code lecture-01-exercise.ipynb

Text searching

- What are some words with similar usage to ‘good’?

```
In [25]: text.similar("good")
```

```
great much large small the in it that long white common old with whale
well certain close such important considerable
```

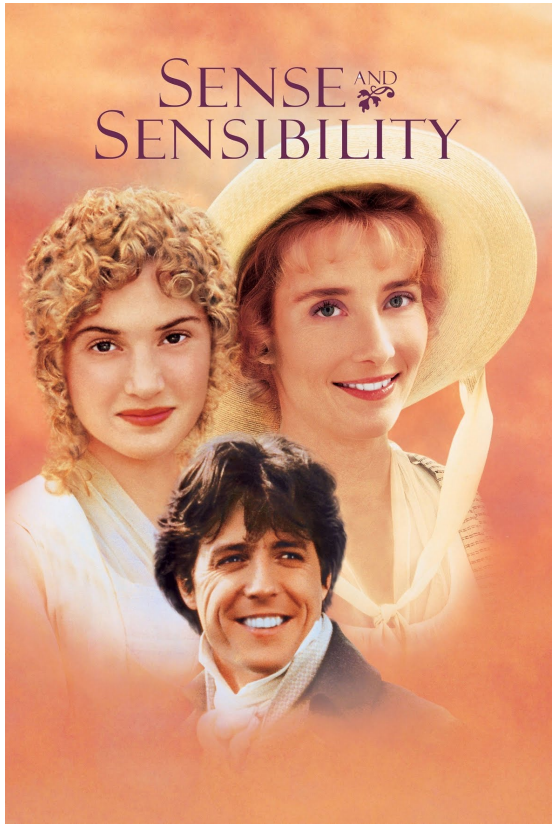
- What are the contexts in which the words ‘good’ and ‘great’ can be used interchangeably?

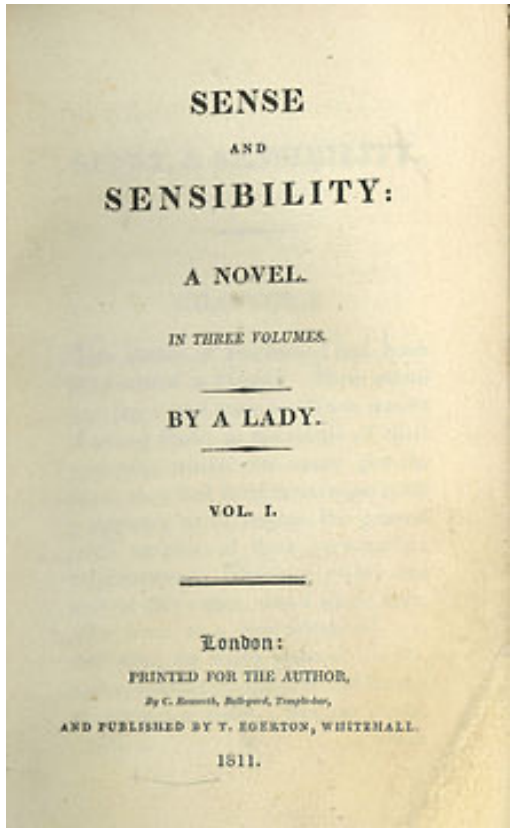
```
In [26]: text.common_contexts(['good','great'])
```

```
a_deal as_a a_long a_christian a_man a_whale so_a the_god too_a
```

Text searching

- Plotting the distribution of specific words in the text: the *text2* corpus, from the novel *Sense and Sensibility*. Elinor, Marianne, Edward, and John are four key characters in the novel—how do they appear throughout the novel?



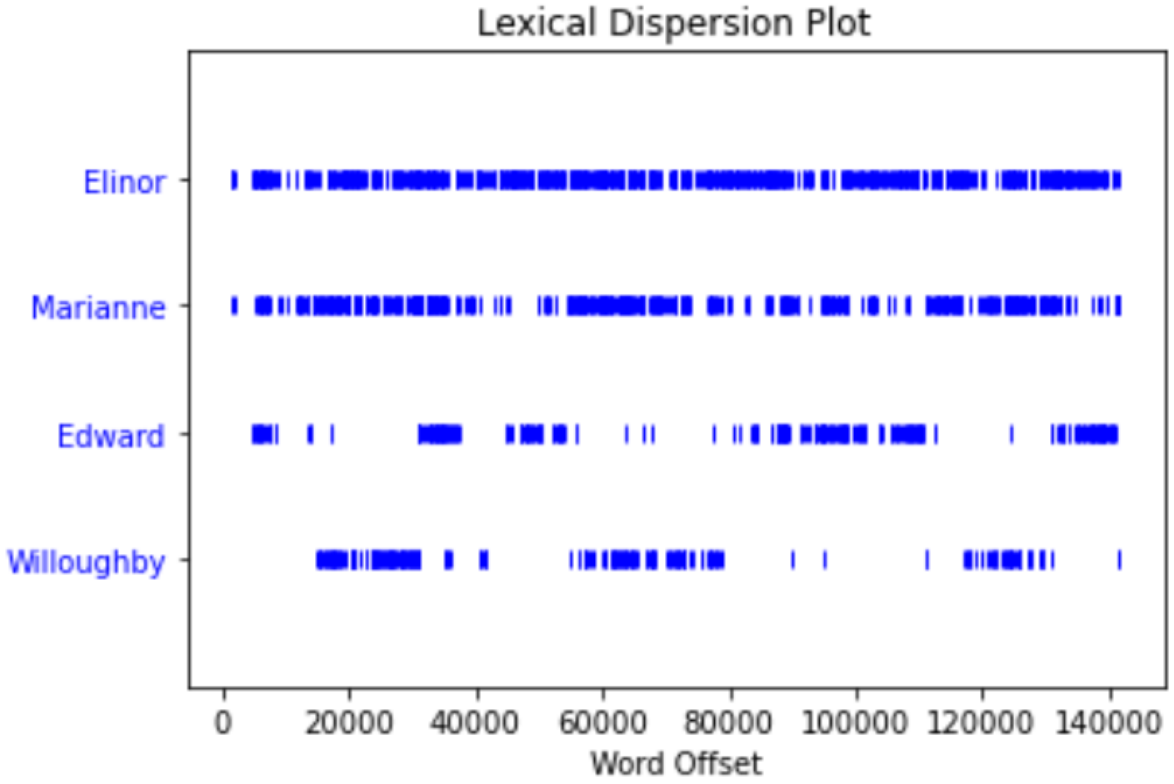

SENSE
AND
SENSIBILITY:
A NOVEL.
IN THREE VOLUMES.
BY A LADY.
VOL. I.
London:
PRINTED FOR THE AUTHOR,
AND PUBLISHED BY T. EGERTON, WHITEHALL.
1811.

```
In [27]: text2
Out[27]: <Text: Sense and Sensibility by Jane Austen 1811>
```

- Elinor Dashwood
- Marianne Dashwood
- Edward Ferrars
- John Willoughby

Text searching

```
text2.dispersion_plot(['Elinor', 'Marianne', 'Edward', 'Willoughby'])
```



Marianne Elinor

Willoughby

Edward

Regular expressions

- **A formal language for specifying text strings**
- **How can we search for any of these?**
  - woodchuck (土拨鼠)
  - woodchucks
  - Woodchuck
  - Woodchucks
- **Need a tool to help us**

lecture-01-exercise.ipynb

RE: disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

```
import re

str1 = "This string contains Woodchuck and woodchuck."
result = re.search(pattern=r"[wW]oodchuck", string=str1)
print(result)
result = re.search(pattern=r"[wW]ooodchuck", string=str1)
print(result)
```

- Ranges [A-Z]

Pattern	Matches	Example String
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

RE: negation in disjunction

- Negations: e.g., [^Ss]
- **Caret** (脱字符): negation only when first in []

Pattern	Matches	Example String
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[^e^]	Neither e nor ^	Look here
a\^b	The pattern a^b	Look up a^b now

```
import re

str1 = "Look up a^b now"
result = re.search(pattern=r"a\^b", string=str1)
print(result)
```

```
<re.Match object; span=(8, 11), match='a^b'>

Process finished with exit code 0
```

RE: more disjunction

- **Woodchucks is another name for groundhog!**
- **The pipe | for disjunction**

Pattern	Matches
groundhog|woodchuck	groundhog and woodchuck
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	**???**

RE: ? * + .

Pattern	Matches	Matched examples
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+	1 or more of previous char	baa baaa baaaa baaaaa
beg.n	any char	begin begun begun beg3n

```
str1 = "oh! ooh!  oooh! ooooh!"
result = re.search(pattern=r"oo+h!", string=str1)
print(result)
```

```
<re.Match object; span=(4, 8), match='ooh!'>
```

RE: find all “the” in a text

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

  the

  Misses capitalized examples

  [tT]he

  Incorrectly returns other or theology

  [^a-zA-Z][tT]he[^a-zA-Z]

- More on Regular Expression: Chapter 3 on Natural Language Processing with Python
  http://www.nltk.org/book/ch03.html

Outline

- Course introduction
- Basics for Python, nltk, spacy
- **Tokenization**
- Minimum edit distance

Tokenization

- Tokenization is the process of breaking down text into pieces, called tokens, such as words, subwords, or characters, which serve as the input for the model, enabling it to learn effective models.
  - Input text: What is tokenization?
  - Output token IDs: 4827 382 6602 2860 30

https://platform.openai.com/tokenizer (GPT-4o)

- There are three types:
  - Word tokenization
  - **Subword tokenization**
  - Character tokenization

https://huggingface.co/docs/transformers/tokenizer_summary

Subword tokenization

- **Idea**: Instead of white-space or single-character segmentation. One can use the raw text data to tell us how to tokenize words. Because tokens can be parts of words as well as whole words.
- Three common algorithms
  - **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
  - **Unigram** (Kudo, 2018)
  - **WordPiece** (Schuster and Nakajima, 2012)
- All subword tokenization algorithms have 2 parts
  - A **token learner** that takes a raw training corpus and induces a vocabulary (a set of tokens). Most subword algorithms are run inside space-separated tokens. So we commonly first add a special end-of-word symbol "_" before space in training corpus.
  - A **token segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

Subword tokenization - BPE

- BPE token learner
  - Set vocabulary $V = \{A, B, C, D, \ldots, a, b, c, d \ldots.\}$
  - 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.

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

Subword tokenization - BPE

- Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new
- **Step 0**: Add a special end-of-word symbol "_" (word boundaries) before space in training corpus and then tokenize text by whitespace. We then separate into letters.
- **Step 1**: Create the vocabulary: {_, d, e, i, l, n, o, r, s, t, w}.
- **Step 2**: Repeat the merging steps, two symbols that are most frequently adjacent in the training corpus.

Subword tokenization - BPE

- Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new
- Add end-of-word tokens, resulting in this vocabulary:

Vocabulary

_, d, e, i, l, n, o, r, s, t, w

Corpus representation

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

Subword tokenization - BPE

- **BPE token learner**

Vocabulary

_, d, e, i, l, n, o, r, s, t, w

Corpus representation

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

Merge e r to er

Vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

Corpus representation

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

Subword tokenization - BPE

- **BPE token learner**

Vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

Corpus representation

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

Merge er _ to er_

Vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

Corpus representation

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

Subword tokenization - BPE

- **BPE token learner**

Vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

Corpus representation

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

Merge n e to ne

Vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_, ne

Corpus representation

5 l o w _
2 l o w e s t _
6 ne w er_
3 w i d er_
2 ne w _

Subword tokenization - BPE

- **BPE token learner**

- The next merges are:

Merge	**Current Vocabulary**
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, low, newer_
(low, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, low, newer_, low_

Subword tokenization - BPE

- **Token segmenter:** On the test data, run each merge learned from the training data:
  - Greedily and in the order we learned them
  - Test frequencies don't play a role
- So, merge every e r to er, then merge er _ to er_, etc.
- **Testing examples**
  - Test set "n e w e r _" would be tokenized as a full word
  - Test set "l o w e r _" would be two tokens: "low er_"

- **One nice property of BPE is that tokens usually include frequent words and frequent subwords which are often morphemes like -est or -er.**
- For example, unlikeliest has 3 morphemes un-, likely, and -est.

- **Question: Given N characters of training corpus and k merges, what is the best time complexity to do BPE tokenization?**

Zouhar, Vilém, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. "A Formal Perspective on Byte-Pair Encoding." In Findings of the Association for Computational Linguistics: ACL 2023, pp. 598-614. 2023.

Try BPE by yourself

- https://platform.openai.com/tokenizer
- Text = “Chapters 5 to 8 teach the basics of 🤗 Datasets and 🤗 Tokenizers before diving into classic NLP tasks. By the end of this part, you will be able to tackle the most common NLP problems by yourself. By the end of this part, you will be ready to apply 🤗 Transformers to (almost) any machine learning problem! E=mc^2. f(x) = x^2+y^2, print('hello world!’) baojianzhou. asdasfasdgasdg”
  - Tokenizers has been split into Token izers
  - baojianzhou has been split into bao jian zhou
- Another great tool: https://tiktokenizer.vercel.app/
- https://github.com/openai/tiktoken

Outline

- Course introduction
- Basics for Python, nltk, spacy
- Tokenization
- **Minimum edit distance**

How similar are two strings?

- **Spell correction**
  - The user typed "graffe". Given candidate set {graf graft grail giraffe}
  - Which is closest?
- **Computational Biology**
  - Align two sequences of nucleotides

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

  - Resulting alignment

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

How similar are two strings?

- **Machine Translation (Task: Chinese -> English)**
  - Chinese: 这个机场的安全工作由以色列方面负责
  - Reference: Israeli officials are responsible for airport security
  - Model 1: Israeli officials responsibility of airport safety
  - Model 2: Israel is responsible for safety work at this airport
  - Model 3: Israel presides over the security of the airport
  - Model 4: Israel took charge of the airport security

Which translation model is the best?

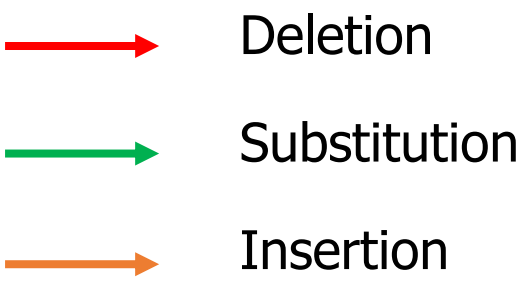
Edit Distance

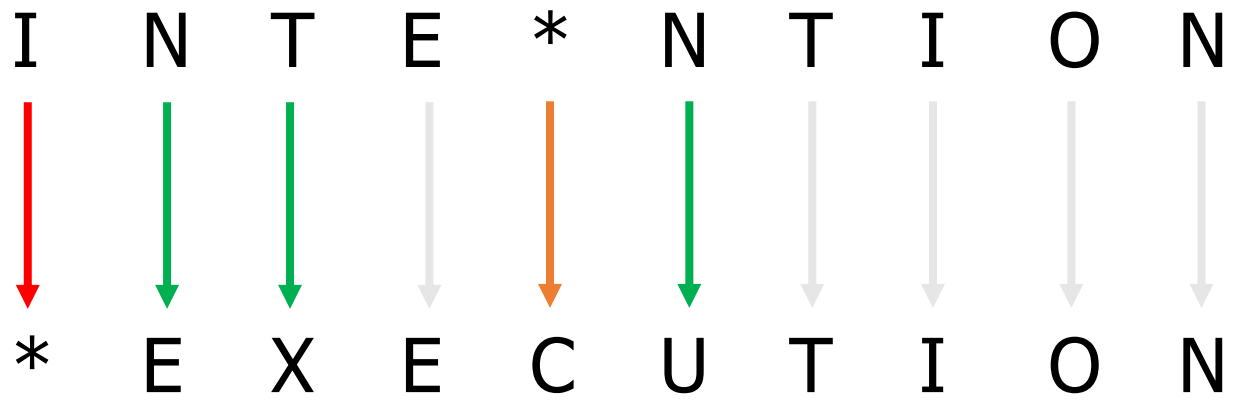
- **The minimum edit distance between two strings**
  - String X -> String Y
- **Is the minimum number of editing operations**
  - Insertion (i)
  - Deletion (d)
  - Substitution (s)
- **Transform String X into String Y**

Minimum Edit Distance (MED)

- Two strings and their **alignment**
- String X = INTENTION
- String Y = EXECUTION

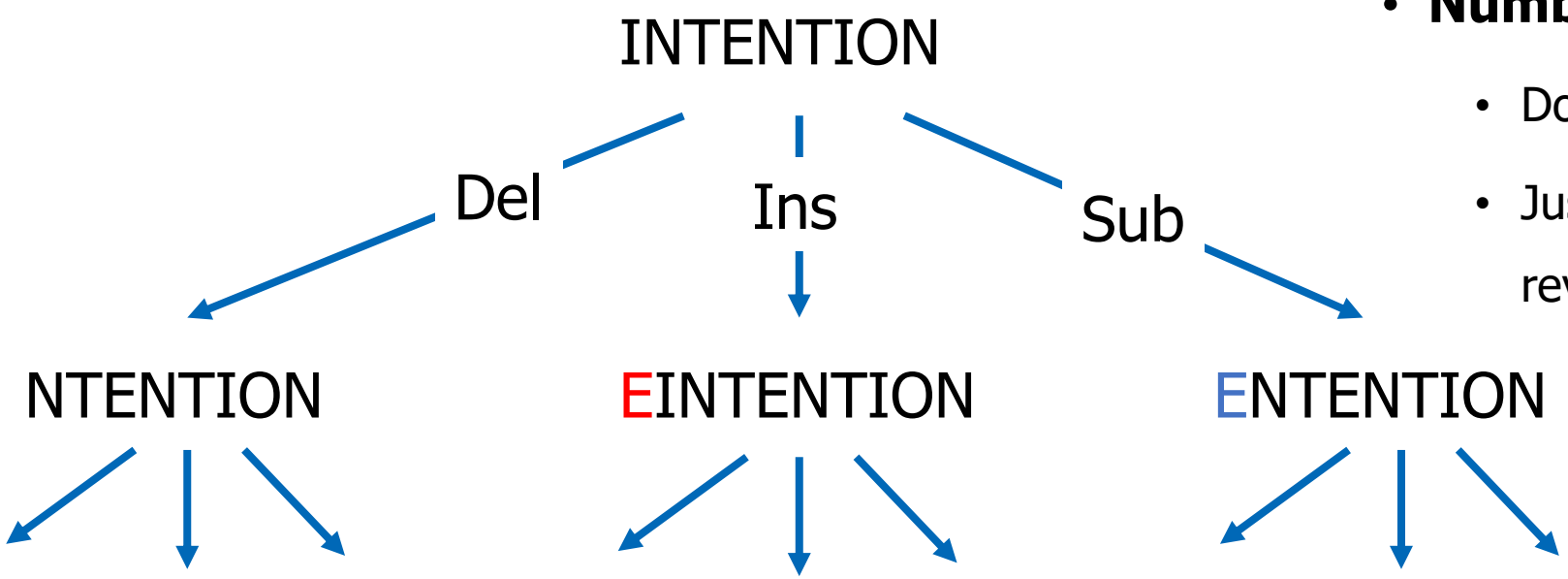
- If each operation has cost of 1
  - Distance between X and Y is 5
- If substitutions cost 2
  - Distance between X and Y is 8





How to find MED?

- Searching for a path (sequence of edits) from the start string to the final string:
  - **Initial state**: the word we are transforming
  - **Operators**: insert, delete, substitute
  - **Goal state**: the word we are trying to get to
  - **Path cost**: the number of weighted edits



- **Number of edit sequences is huge**
  - Do not have to track of all of them
  - Just the shortest path to each of those revisited states.

Defining MED

- For two strings, define
  - X of length n: $X = [x_1, x_2, \ldots, x_n]$
  - Y of length m: $Y = [y_1, y_2, \ldots, y_m]$
- We define $D[i, j]$ the MED between $X[1 \ldots i]$ and $Y[1 \ldots j]$
    - i: the first i characters of X
    - j: the first j characters of Y
- **The edit distance between X and Y is thus $D[n, m]$**

Dynamic programming for MED

- **Initialization**
  - $D[i,j] = j, \text{for } i = 0$
  - $D[i,j] = i, \text{for } j = 0$
- **Recurrence Relation**

For each $i = 1,2,\dots,n$

For each $j = 1,2,\dots,m$

$$D[i,j] = \min \begin{cases} D[i-1,j] + \text{Del}(x_i) \\ D[i,j-1] + \text{Ins}(y_i) \\ D[i-1,j-1] + \text{Sub}(x_i, y_i) \end{cases}$$

Termination return $D[n,m]$ as MED

- **Levenshtein Distance**
  - $\text{Del}(x_i) = 1$
  - $\text{Ins}(y_i) = 1$
  - $\text{Sub}(x_i, y_i) = \begin{cases} 2 & x_i \neq y_i \\ 0 & x_i = y_i \end{cases}$

MED table: initialization

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$i = 0$

$j = 0$

MED table: second column

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1	?								
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$i = 1$

$j = 1$

$$D[i,j] = \min \begin{cases} D[i-1,j] + Del(x_i) \\ D[i,j-1] + Ins(y_i) \\ D[i-1,j-1] + Sub(x_i, y_i) \end{cases}$$

MED table: second column

	N	9								
	O	8								
	I	7								
	T	6								
	N	5								
	E	4								
	T	3								
	N	2								
$i = 1$	I	1	2							
	#	0	1	2	3	4	5	6	7	8	9
		#	E	X	E	C	U	T	I	O	N

$j = 1$

$$D[i,j] = \min \begin{cases} D[i-1,j] + Del(x_i) \\ D[i,j-1] + Ins(y_i) \\ D[i-1,j-1] + Sub(x_i, y_i) \end{cases}$$

Substitution

MED table: second column

N	9									
O	8									
I	7									
T	6									
N	5									
E	4	?								
T	3	4								
N	2	3								
I	1	2								
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$i = 4$

$j = 1$

$$D[i,j] = \min \begin{cases} D[i-1,j] + Del(x_i) \\ D[i,j-1] + Ins(y_i) \\ D[i-1,j-1] + Sub(x_i, y_i) \end{cases}$$

MED table: second column

	N	9								
	O	8								
	I	7								
	T	6								
	N	5								
$i = 4$	E	4	3							
	T	3	4							
	N	2	3							
	I	1	2							
	#	0	1	2	3	4	5	6	7	8	9
		#	E	X	E	C	U	T	I	O	N

$j = 1$

$$D[i,j] = \min \begin{cases} D[i-1,j] + Del(x_i) \\ D[i,j-1] + Ins(y_i) \\ D[i-1,j-1] + Sub(x_i, y_i) \end{cases}$$

Unchanged

Final MED table

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Computing alignment

- Edit distance is not sufficient
  - Need to **align** each character of the two strings to each other
- Keep a "backtrace"
  - Every time we enter a cell, remember where we came from
- When we reach the end,
  - Trace back the path from the upper right to read off the alignment

Computing alignment

$i = 9$

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	**E**	**X**	**E**	**C**	**U**	**T**	**I**	**O**	**N**

$j = 9$

Start at end; at each step, ask which **predecessor** gave the minimum?

1. $D[i-1, j-1] + Sub(x_i, y_i) == D[i, j]$
2. $D[i-1, j] + Del(x_i) == D[i, j]$
3. $D[i, j-1] + Ins(y_i) == D[i, j]$

Computing alignment

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	**E**	**X**	**E**	**C**	**U**	**T**	**I**	**O**	**N**

$i = 5$

$j = 5$

Start at end; at each step, ask which **predecessor** gave the minimum?

1. $D[i-1, j-1] + Sub(x_i, y_i) == D[i, j]$
2. $D[i-1, j] + Del(x_i) == D[i, j]$
3. $D[i, j-1] + Ins(y_i) == D[i, j]$

Computing alignment

	N	9	8	9	10	11	12	11	10	9	8
	O	8	7	8	9	10	11	10	9	8	9
	I	7	6	7	8	9	10	9	8	9	10
	T	6	5	6	7	8	9	8	9	10	11
	N	5	4	5	6	7	8	9	10	11	10
$i = 4$	E	4	3	4	5	6	7	8	9	10	9
	T	3	4	5	6	7	8	7	8	9	8
	N	2	3	4	5	6	7	8	7	8	7
	I	1	2	3	4	5	6	7	6	7	8
	#	0	1	2	3	4	5	6	7	8	9
		#	E	X	E	C	U	T	I	O	N

$j = 4$

Start at end; at each step, ask which **predecessor** gave the minimum?

1. $D[i-1, j-1] + Sub(x_i, y_i) == D[i, j]$
2. $D[i-1, j] + Del(x_i) == D[i, j]$
3. $D[i, j-1] + Ins(y_i) == D[i, j]$

Computing alignment

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	**E**	**X**	**E**	**C**	**U**	**T**	**I**	**O**	**N**

$i = 4$

$j = 3$

Start at end; at each step, ask which **predecessor** gave the minimum?

1. $D[i-1, j-1] + Sub(x_i, y_i) == D[i, j]$
2. $D[i-1, j] + Del(x_i) == D[i, j]$
3. $D[i, j-1] + Ins(y_i) == D[i, j]$

Computing alignment

	N	9	8	9	10	11	12	11	10	9	8
	O	8	7	8	9	10	11	10	9	8	9
	I	7	6	7	8	9	10	9	8	9	10
	T	6	5	6	7	8	9	8	9	10	11
	N	5	4	5	6	7	8	9	10	11	10
	E	4	3	4	5	6	7	8	9	10	9
	T	3	4	5	6	7	8	7	8	9	8
	N	2	3	4	5	6	7	8	7	8	7
$i = 1$	I	1	2	3	4	5	6	7	6	7	8
	#	0	1	2	3	4	5	6	7	8	9
		#	E	X	E	C	U	T	I	O	N

$j = 0$

Start at end; at each step, ask which **predecessor** gave the minimum?

1. $D[i-1, j-1] + Sub(x_i, y_i) == D[i, j]$
2. $D[i-1, j] + Del(x_i) == D[i, j]$
3. $D[i, j-1] + Ins(y_i) == D[i, j]$

Computing alignment

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

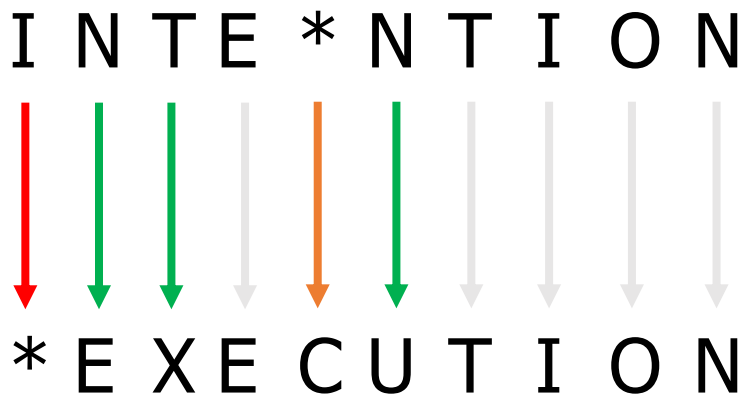
$i = 0$

$j = 0$

Start at end; at each step, ask which **predecessor** gave the minimum?

1. $D[i-1, j-1] + Sub(x_i, y_i) == D[i, j]$
2. $D[i-1, j] + Del(x_i) == D[i, j]$
3. $D[i, j-1] + Ins(y_i) == D[i, j]$

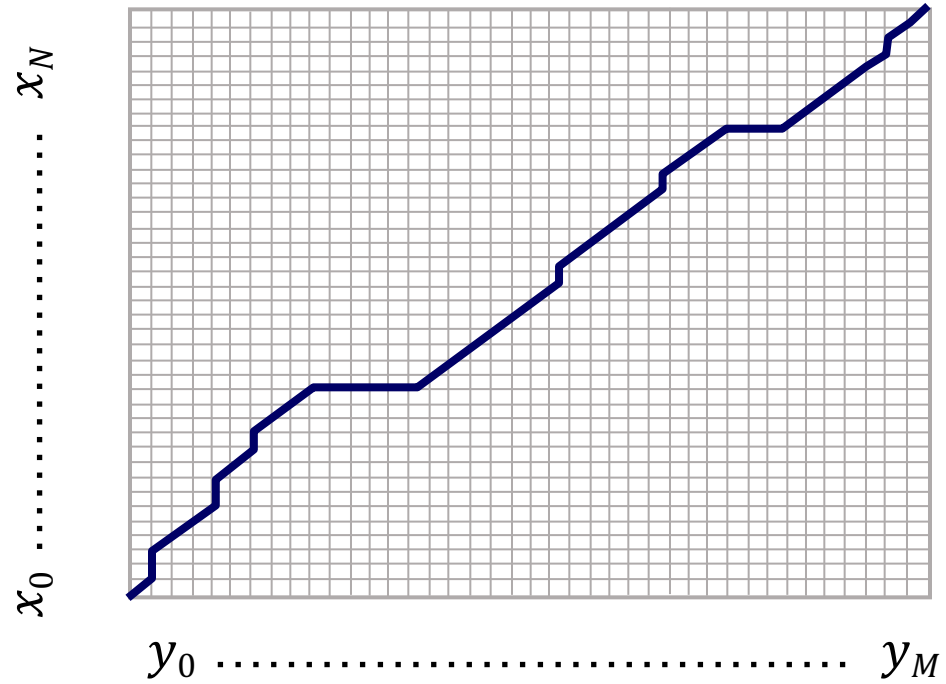
When $(i, j) == (0,0)$ Stop



Computing alignment - properties

When entering a value in each cell, we mark which of the three neighboring cells we came from with up to three arrows.

	#	e	x	e	c	u	t	i	o	n
#	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8	← 9
i	↑ **1**	↖←↑ 2	↖←↑ 3	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖ 6	← 7	← 8
n	↑ 2	↖←↑ **3**	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↑ 7	↖←↑ 8	↖ 7
t	↑ 3	↖←↑ 4	↖←↑ **5**	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖ 7	←↑ 8	↖←↑ 9	↑ 8
e	↑ 4	↖ 3	← 4	↖← **5**	← **6**	← 7	←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9
n	↑ 5	↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ **8**	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖↑ 10
t	↑ 6	↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖ **8**	← 9	← 10	←↑ 11
i	↑ 7	↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9	↖ **8**	← 9	← 10
o	↑ 8	↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↑ 10	↑ 9	↖ **8**	← 9
n	↑ 9	↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖←↑ 12	↑ 11	↑ 10	↑ 9	↖ **8**



- Every non-decreasing path from $(0,0)$ to (n,m) corresponds to an alignment of the two sequences.
- An optimal alignment is composed of optimal subalignments.
- Algo. time: $O(nm)$ space: $O(nm)$
- Backtrace time and space: $O(n+m)$

Exercises and readings

1. Register a ChatGPT account if you have not done yet, see instruction: https://www.yanlutong.com/gonglue/30511.html
2. For NLP and Python beginners: Alice Zhao, Natural Language Processing in Python, https://www.youtube.com/watch?v=xvqsFTUsOmc&t=1607s
3. Implement MED and backtrack algorithm
4. Read and practice
   - https://www.nltk.org/book/ch01.html
   - https://www.nltk.org/book/ch02.html
5. **Next lecture**: language model

References

- [1] 张奇、桂韬、黄萱菁，自然语言处理导论，https://intro-nlp.github.io/, Chapter 1, 2023.
- [2] https://learn.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/natural-language-processing
- [3] https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html
- [4] Transformer model timeline: https://ai.v-gar.de/ml/transformer/timeline/
- [5] Dan’s book, Chapter 2.
- [6] Dan’s slides