Big Model Systems and Application

Zhiyuan Liu

liuzy@tsinghua.edu.cn

THUNLP

Course Info

- Big Model Systems and Application

- Summer course, practice course
- First Year at Tsinghua

- Content
  - Lecture: 3 weeks, 3 classes per week, with several exercises
  - Practice: 1 week for project, one final project on the big model topics

What to Teach

- Big picture knowledge about big models in NLP and beyond
  - Basics of NLP and neural models
  - Why and how NLP models become larger
  - New paradigms and methods in big models
  - Open problems and challenges in big model systems
- Ability to use open-source toolkits to build practical systems based on big models
  - Utilization of big models is not easy due to the giant size
  - Learn to use open-source toolkits
- Ability to solve novel open problems with the power of big models
  - How to review related works in the related area
  - How to identify the key challenges for the task
  - How to figure out solutions to the challenge from big models' point of view

About Me

- Zhiyuan Liu
- Research Interests
  - Natural Language Processing, Big Models
- Contact Info
  - Email: liuzy@tsinghua.edu.cn
  - Office: Room 4-506, FIT Building
- Webpage
  - nlp.csai.tsinghua.edu.cn/~lzy/

Course Plan

Basic Knowledge of Big Models

- L1 - NLP & Big Model Basics (GPU server, Linux, Bash, Conda, …)
- L2 - Neural Network Basics (PyTorch)
- L3 - Transformer and PLMs (Huggingface Transformers)

Key Technology of Big Models

- L4 - Prompt Tuning & Delta Tuning (OpenPrompt, OpenDelta)
- L5 - Efficient Training & Model Compression (OpenBMB suite)
- L6 - Big-Model-based Text understanding and generation

Interdisciplinary Application of Big Models

- L7 - Big Models X Biomedical Science
- L8 - Big Models X Legal Intelligence
- L9 - Big Models X Brain and Cognitive Science

Assignments and Grading Policy

- PF, one exercise per class (9 in total)
- Course Project (will be announced at the end of 3rd class)
  - Basic Projects, for students with busy schedules
    - ➢ Apply big models to NLP tasks or Interdisciplinary Fields
  - **Research Projects**, for self-motivated students
    - ➢ About 14 research projects, and choose the one you are interested in
    - ➢ Supervised by experienced graduate students in our group
    - ➢ Expect research achievements including papers, open-source tools, or demos, etc.

TA Info

- Lead TA: Shengding Hu
- Master student
- Research Interest: Pre-trained Models
- Email: hsd20@mails.tsinghua.edu.cn

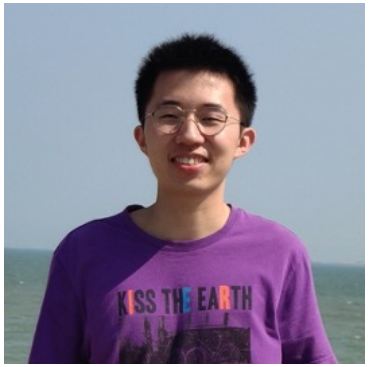
- TA: Shi Yu
- PhD student
- Research Interest: Information Retrieval & Question Answering

- TA: Weize Chen
- PhD student
- Research Interest: Architecture of Big Models

TA Info

- TA: Yujia Qin
- PhD student
- Research Interest: Pre-trained Models

- TA: Zhengyan Zhang
- PhD student
- Research Interest: Pre-trained Models



- TA: Ganqu Cui
- PhD student
- Research Interest: NLP Safety

TA Info

- TA: Chaojun Xiao
- PhD student
- Research Interest: Legal Intelligence

- TA: Zheni Zeng
- PhD student
- Research Interest: Biomedical Natural Language Processing

Other lecturers will be introduced in corresponding classes.

Course Resources

- WeChat Group: Please contact TA to join
- GitHub
  - https://github.com/thunlp/BMCourse
  - For QA
  - For exercises
  - For other resources
- Paracloud Computational Resources

Lecture 1
NLP and Big Model Basics

THUNLP

Content

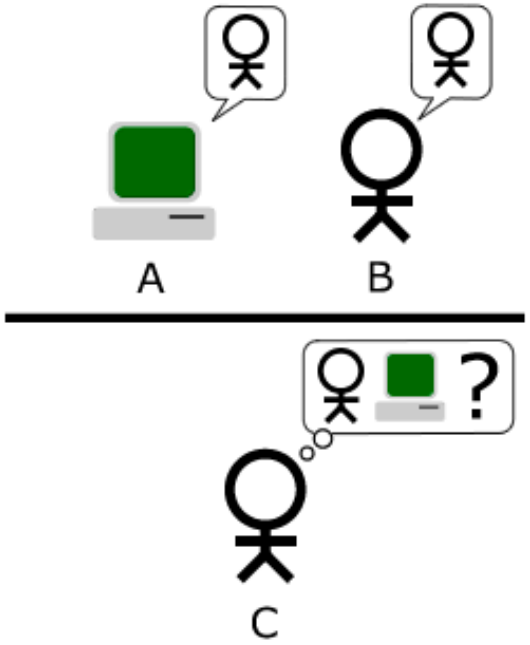
- NLP Basics
  - Why is NLP Important?
  - Typical Tasks & Applications
  - Basic Concepts 1: Distributed Word Representation
  - Basic Concepts 2: Neural Language Models
- Big Models Basics
  - The Trip to Big Models
  - Why are Big Models Important?
  - Paradigms behind Big-models
  - Typical Cases of Big-models
  - Big Model Demos
- Application Basics
  - Coding Environment & GPU Server

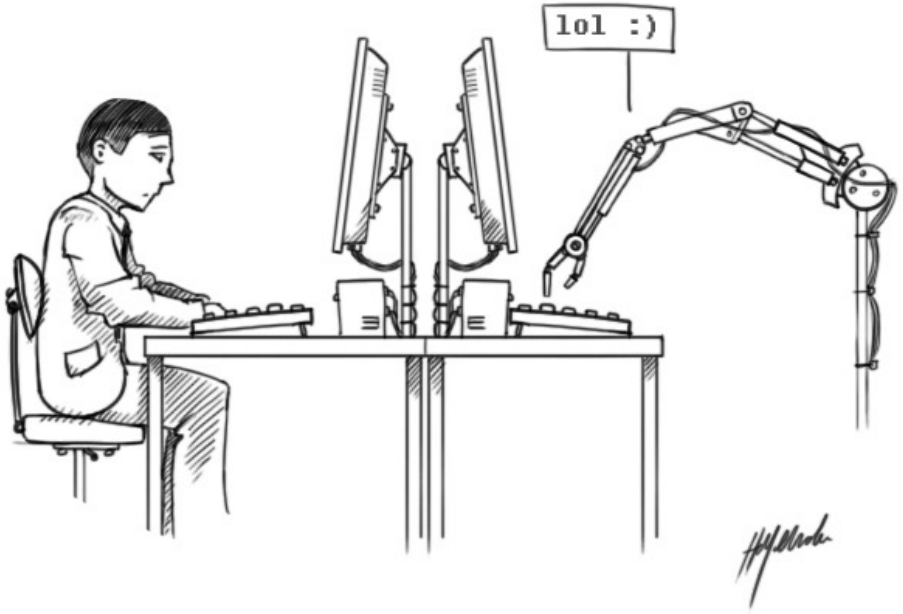
NLP Basics
Why is NLP Important?

THUNLP

Scientific Impact of NLP

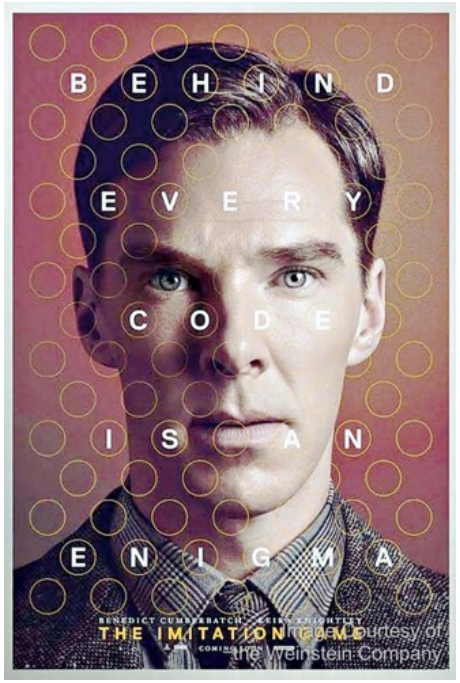
- Turing Test: A test of machine ability to exhibit intelligent behavior indistinguishable from a human
- Language is the communication tool in the test

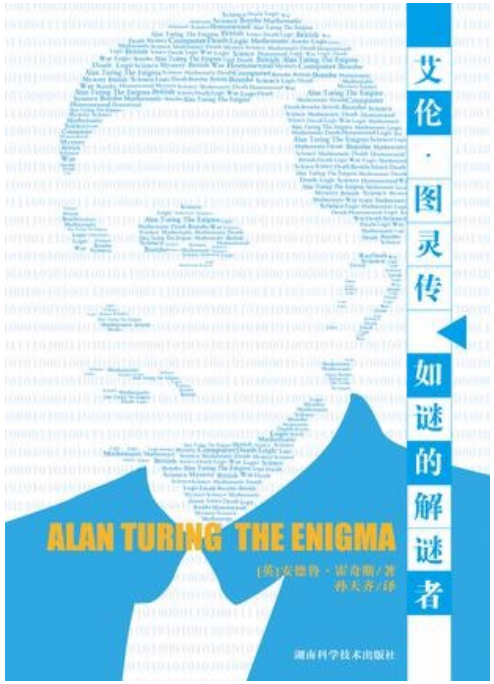


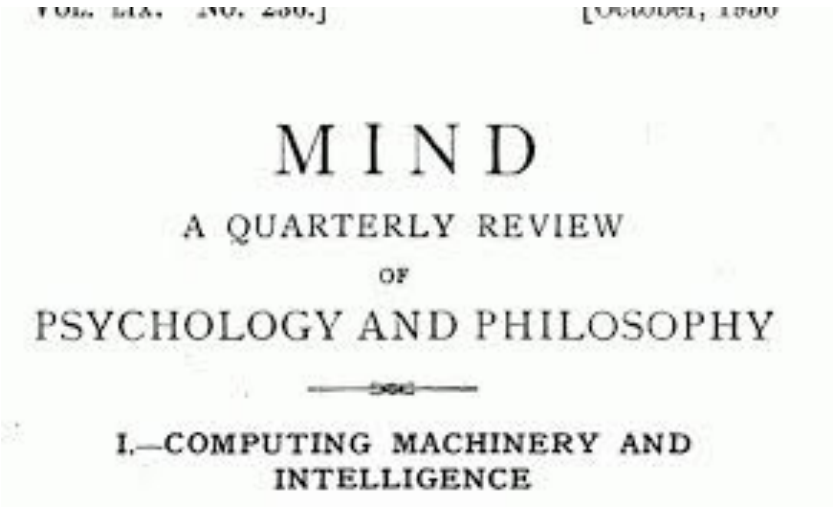


Scientific Impact of NLP

- Origin Version: Imitation Game







MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

Scientific Impact of NLP

- Natural language question-answering
- 2011: IBM Watson DeepQA system competed on Jeopardy! and received the first place
- A new milestone of AI after DeepBlue won world champion of chess in 1997

Q: Who was presidentially pardoned on September 8, 1974?
A: Nixon.

Scientific Impact of NLP

- Many institutes take NLP as key research areas

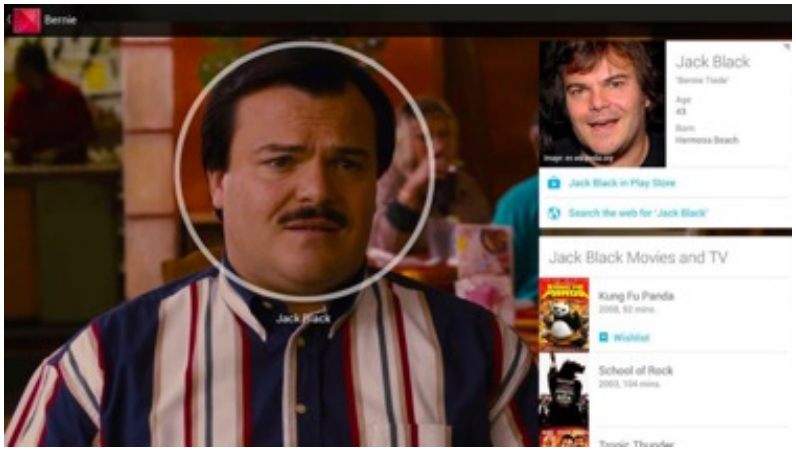
Application Impact of NLP

- IT giants launch their NLP products

Apple Siri

Speech Translator

Sogou Input



Google Knowledge Graphs

A Nice Review on NLP

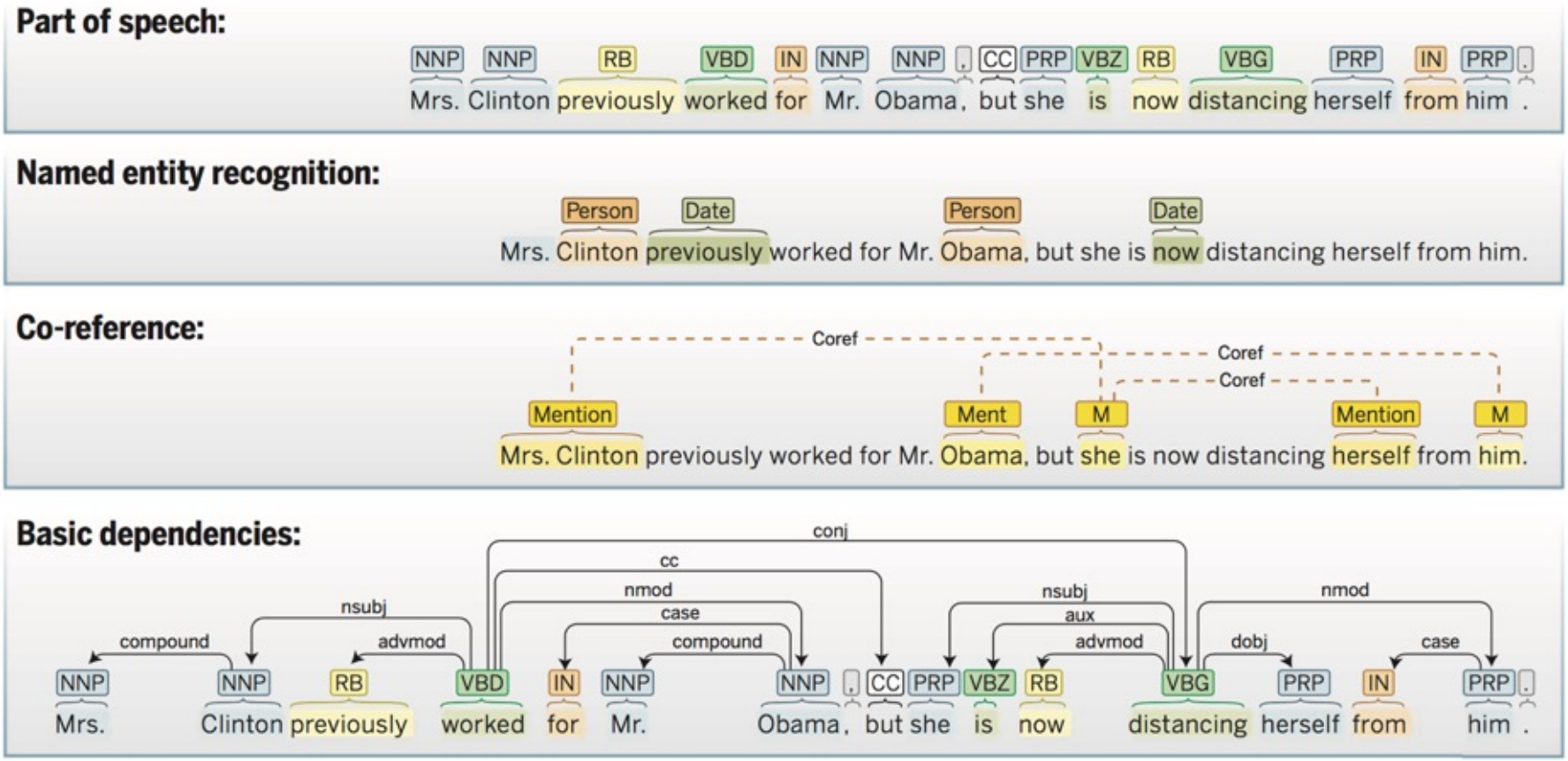
- Advances in Natural Language Processing
- Julia Hirschberg, Columbia University
  - AAAI, ACL Fellow
- Christopher Manning， Stanford University
  - ACM, AAAI, ACL Fellow
  - Google Scholar Citation > 50,000

NLP Basics
Typical Tasks & Applications

THUNLP

NLP Tasks

- Basic Tasks of NLP



Search Engines and Ads

Structural Knowledge

write

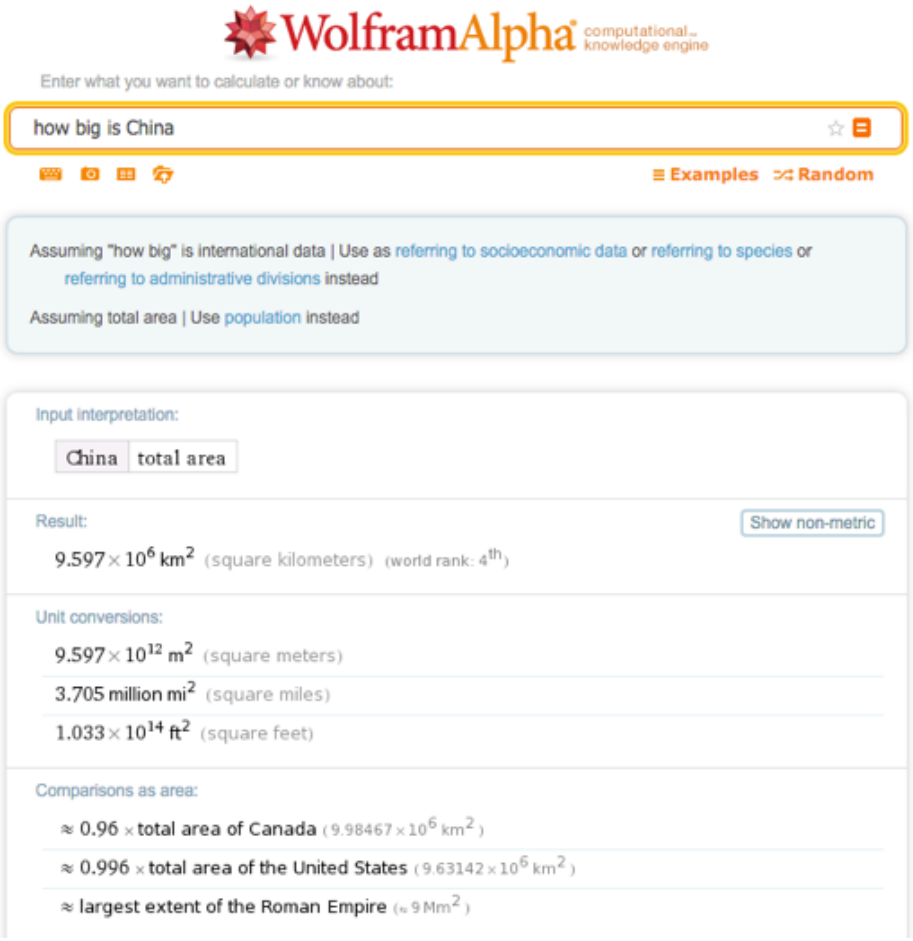
(***William Shakespeare***, book/author/works_written, ***Romeo and Juliet***)

head entity **relation** **tail entity**

Knowledge Graph

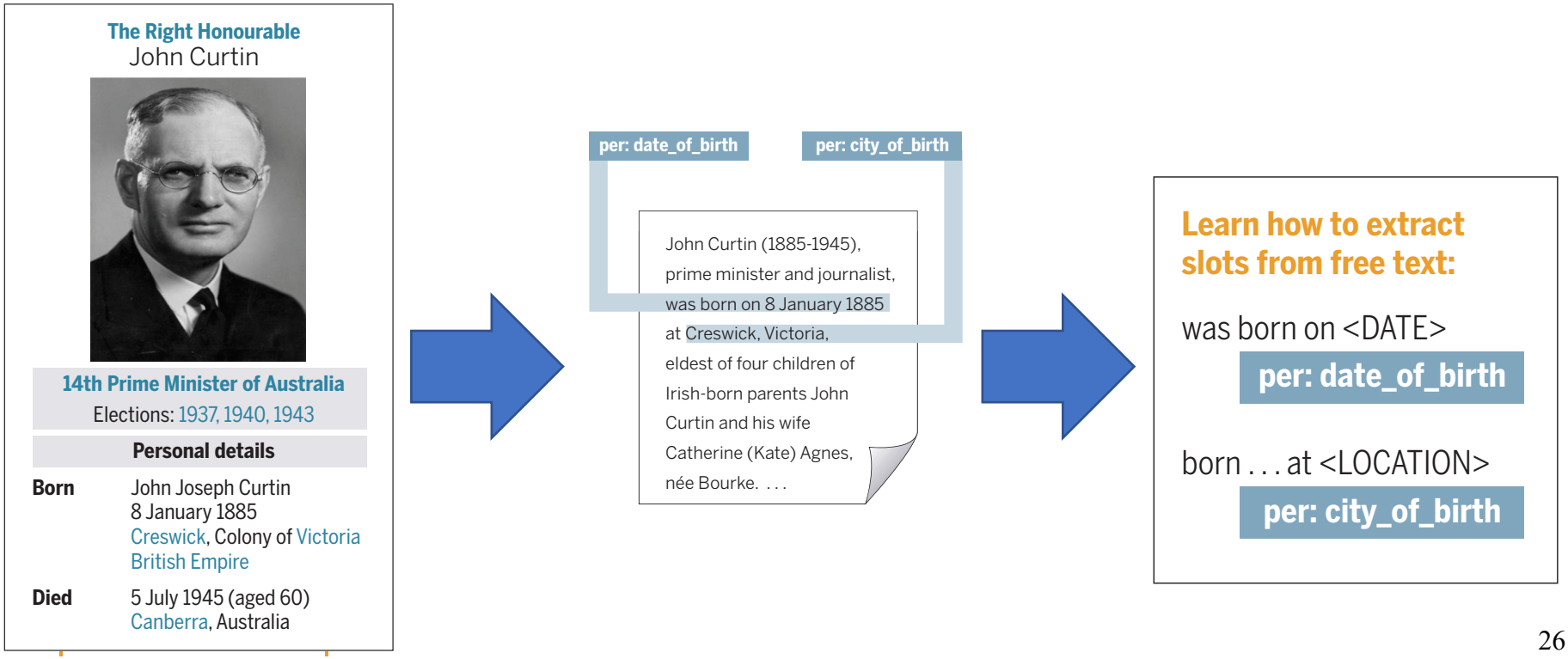
KG Application: Question Answering

- KG provides the knowledge facts of a question



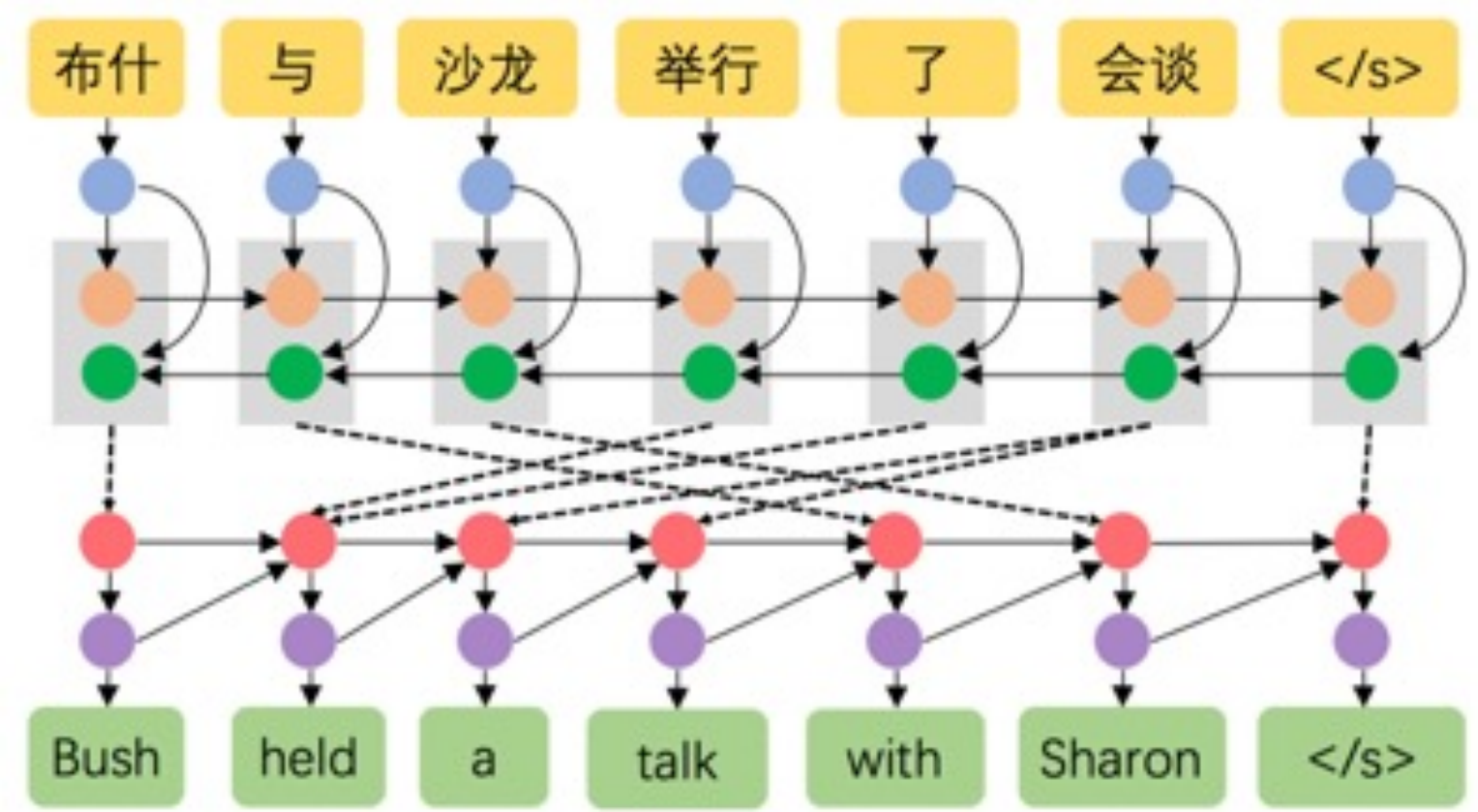
Machine Reading

- Machine reading aims to extract structural knowledge (e.g., relational facts between entities) from plain text
- Can expand and update knowledge graphs



Personal Assistant

Machine Translation

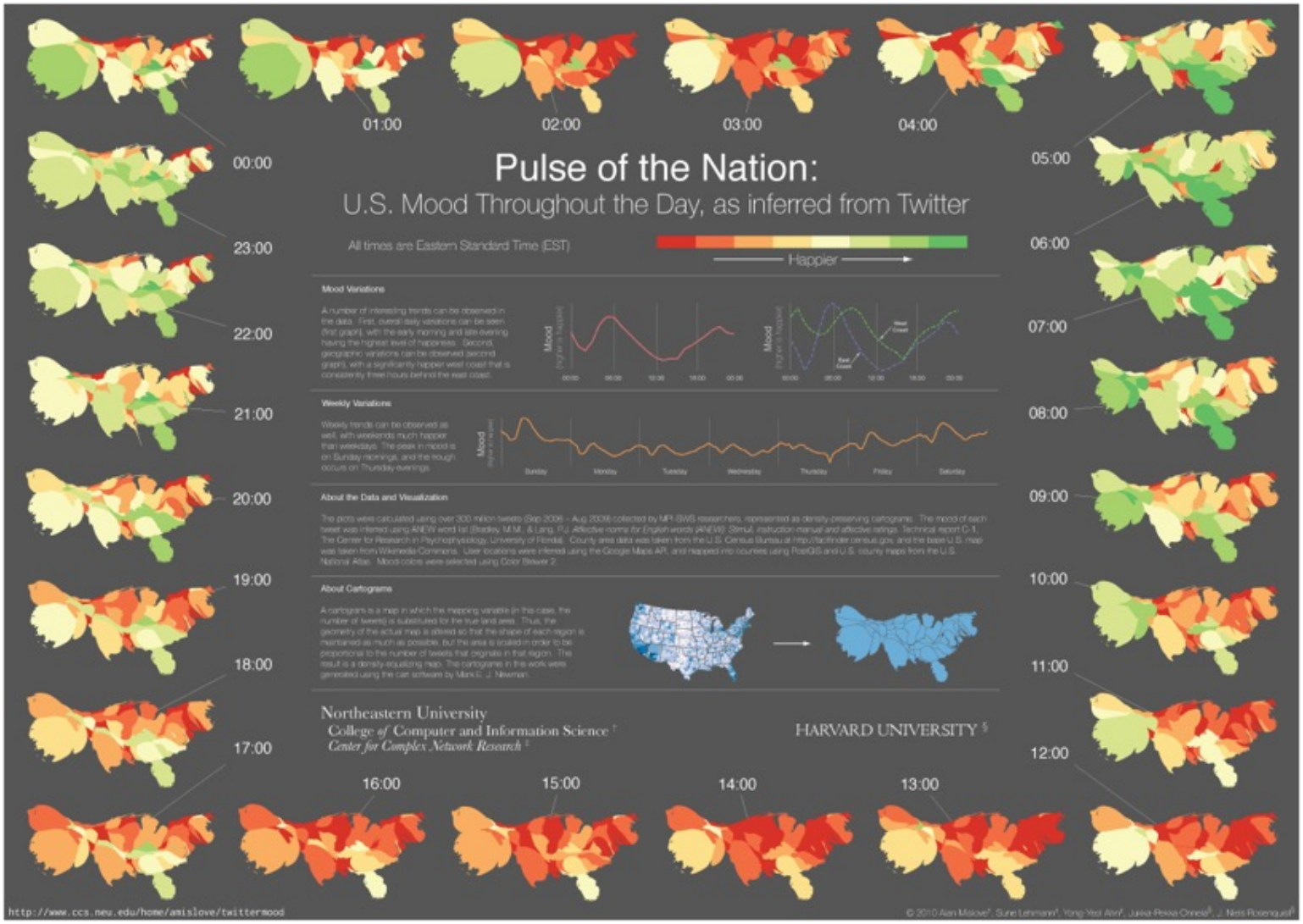


Sentiment Analysis and Opinion Mining

- Infer personal states via text or speech
  - Including opinions, emotions, …

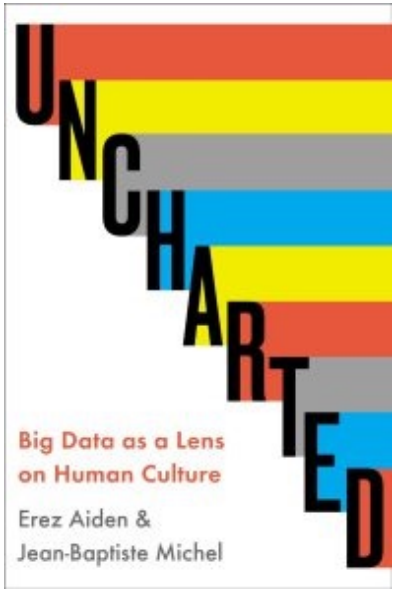
- Detect opinion holders and targets

Sentiment Analysis and Opinion Mining



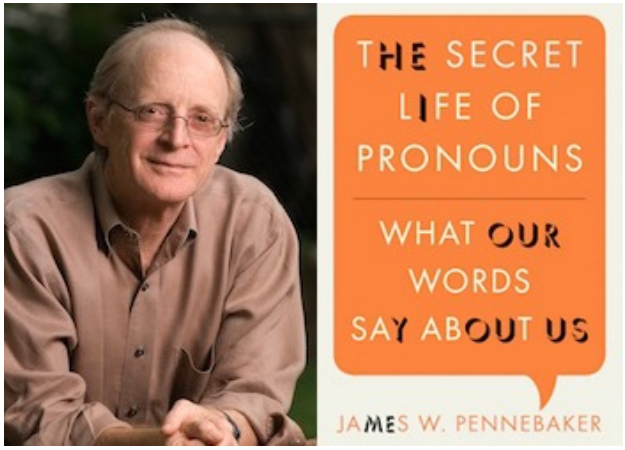
Computational Social Science

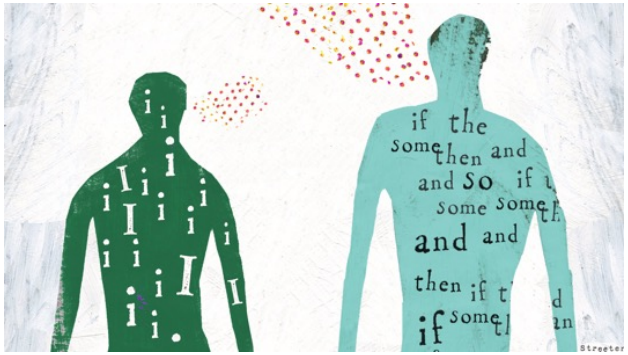
- Culturomics: www.culturomics.org
- Harvard researchers use keywords over Google Books (5 million books from 1800 to 2000) to study the evolution of human culture
- Google Book N-gram: books.google.com/ngrams



Computational Social Science

- Use language usage to study psychology states of humans





LIWC Results

Details of Writer: 40 year old Female
Date/Time: 6 January 2014, 1:02 am

LIWC results from input text

LIWC categories

LIWC Dimension	Your Data	Personal Texts	Formal Texts
Self-references (I, me, my)	8.33	11.4	4.2
Social words	4.17	9.5	8.0
Positive emotions	2.08	2.7	2.6
Negative emotions	1.04	2.6	1.6
Overall cognitive words	3.12	7.8	5.4
Articles (a, an, the)	2.08	5.0	7.2
Big words (> 6 letters)	20.83	13.1	19.6

LIWC results from personal text and formal writing for comparison

The text you submitted was 96 words in length.

Your writing:

I'm newly diagnosed with type 2 diabetes. I also struggle with both calcium and uric acid kidney stones as well as the rare blood disorder LEIDEN FACTOR V. Is there anyone in this community who deals with Leiden as well as diabetes? If there is I would LOVE to be able to chat with you regarding diet and possible weight loss plans. I currently have no regular doctor and a insurance so my diabetes is uncontrolled at this time. I am working hard to educate myself AND make the necessary changes to improve my current health.

Input text: A post from a 40 year old female member in American Diabetes Association online community

NLP Basics
Distributed Word Representation

THUNLP

Word Representation

- Word representation: a process that transform the symbols to the machine understandable meanings

- Definition of meaning (Webster Dictionary)
  - The thing one intends to convey especially by language
  - The logical extension of a word

- How to represent the meaning so that the machine can understand?

Goal of Word Representation

- Compute word similarity
  - WR(Star) ≃ WR(Sun)
  - WR(Motel) ≃ WR(Hotel)
- Infer word relation
  - WR(China) − WR(Beijing) ≃ WR(Japan) - WR(Tokyo)
  - WR(Man) ≃ WR(King) − WR(Queen) + WR(Woman)
  - WR(Swimming) ≃ WR(Walking) − WR(Walk) + WR(Swim)

Synonym and Hypernym

- Use a set of related words, such as synonyms and hypernyms to represent a word
  - e.g. WordNet, a resource containing synonym and hypernym sets.

Synonyms of "Good" in WordNet:
(n)good,goodness
(n)commodity,trade_good,good
(s)full,good
(s)adept,expert,good,practiced,proficient,skillful
(s)estimable,good,honorable,respectable
(s)beneficial,good
…

Hypernyms of "NLP" in WordNet:

```
[Synset('information_science.n.01'),
 Synset('science.n.01'),
 Synset('discipline.n.01'),
 Synset('knowledge_domain.n.01'),
 Synset('content.n.05'),
 Synset('cognition.n.01'),
 Synset('psychological_feature.n.01'),
 Synset('abstraction.n.06'),
 Synset('entity.n.01')]
```

Problems of Synonym/Hypernym Representation

- Missing nuance
  - ("proficient", "good") are synonyms only in some contexts
- Missing new meanings of words
  - Apple (fruit → IT company)
  - Amazon (jungle → IT company)
- Subjective
- Data sparsity
- Requires human labor to create and adapt

One-Hot Representation

- Regard words as discrete symbols
- Word ID or one-hot representation
- E.g.

word	ID	one-hot vector
star	2	[0, 0, 1, 0, 0, 0, …]
sun	3	[0, 0, 0, 1, 0, 0, …]

  - Vector dimension = # words in vocabulary
  - Order is not important

Problems of One-Hot Representation

- $similarity(star, sun) = (v_{star}, v_{sun}) = 0$

- All the vectors are orthogonal. No natural notion of similarity for one-hot vectors.

Represent Word by Context

- The meaning of a word is given by the words that frequently appear close-by
  - *"You shall know a word by the company it keeps."* (J.R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP.
- Use context words to represent stars
  - Co-occurrence Counts
  - Words Embeddings

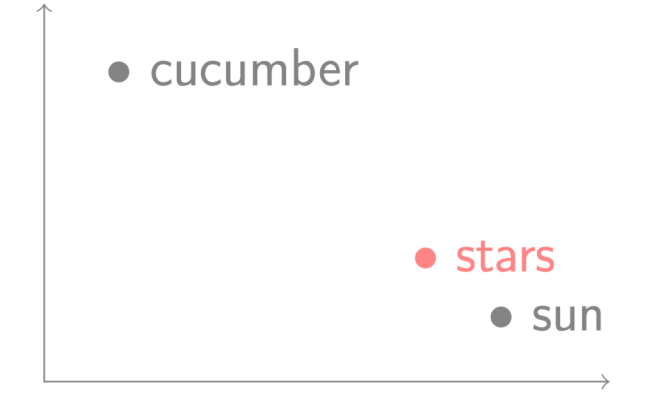
```
he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
 surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold
```

Co-Occurrence Counts

- Count-based distributional representation

he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold

- Term-Term matrix
  - How often a word occurs with another
- Term-Document matrix
  - How often a word occurs in a document



	shining	bright	trees	dark	look
stars	38	45	2	27	12

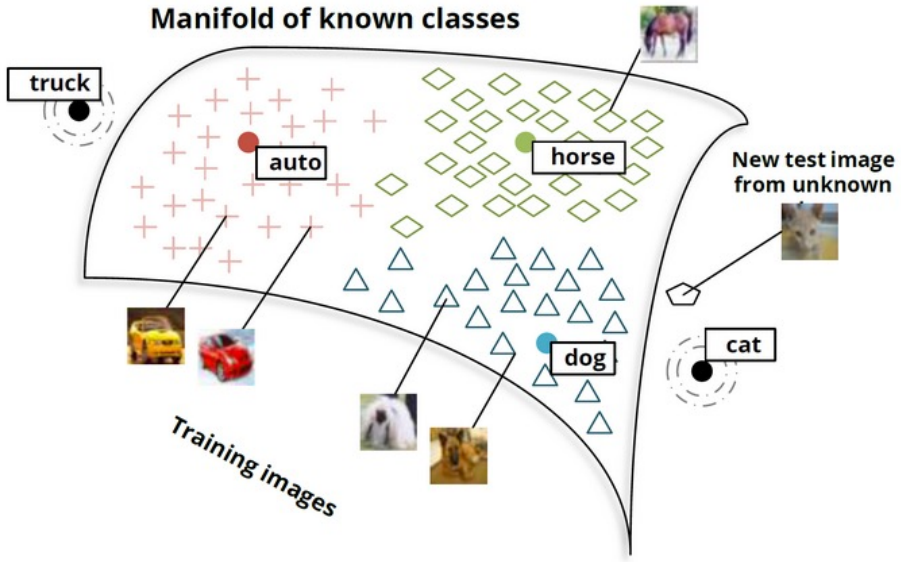
Problems of Count-Based Representation

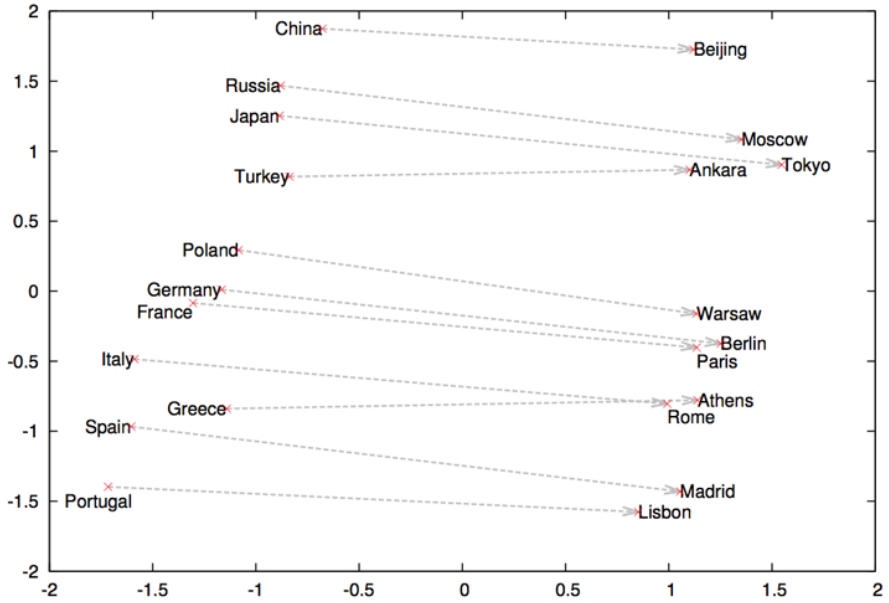
- Increase in size with vocabulary
- Require a lot of storage
- sparsity issues for those less frequent words

  → Subsequent classification models will be less robust

Word Embedding

- Distributed Representation
  - Build a dense vector for each word learned from large-scale text corpora
  - Learning method: Word2Vec (We will learn it in the next class)





NLP Basics
Language Modeling

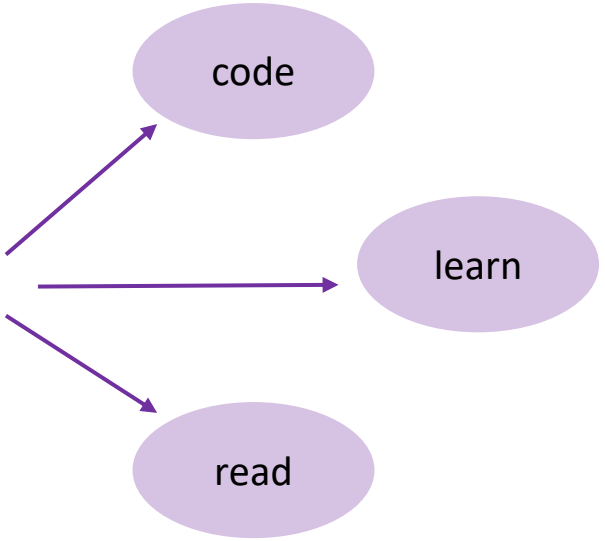
THUNLP

Language Model

- Language Modeling is the task of predicting the upcoming word
  - Compute conditional probability of an upcoming word w_n:

$$P(w_n|w_1, w_2, \cdots, w_{n-1})$$

Never too late to _____



Language Model

- A language model is a probability distribution over a sequence of words
  - Compute joint probability of a sequence of words:

$$P(W) = P(w_1, w_2, \cdots, w_n)$$

  - Compute conditional probability of an upcoming word w_n :

$$P(w_n | w_1, w_2, \cdots, w_{n-1})$$

- How to compute the sentence probability?

Language Model

- Assumption: the probability of an upcoming word is only determined by all its previous words

1. $P(Never, too, late, to, learn) =$

$$P(Never) \times P(too|Never) \times P(late|Never, too) \times$$

$$P(to|Never, too, late) \times P(learn|Never, too, late, to)$$

2. $P(learn|Never, too, late, to) = \frac{P(Never,too,late,to,learn)}{P(Never,too,late,to)}$

- Language Model

$$P(w_1, w_2, \cdots, w_n) = \prod_i P(w_i|w_1, w_2, \cdots, w_{i-1})$$

N-gram Model

- Collect statistics about how frequent different n-grams are, and use these to predict next word.
- E.g., 4-gram

$$P(w_j|never\ to\ late\ to) = \frac{count(too\ late\ to\ w_j)}{count(too\ late\ to)}$$

- Problem:
  - Need to store count for all possible *n*-grams. So model size is $O(\exp(n))$.

N-gram Model

- Markov assumption

$$P(w_1, w_2, \cdots, w_n) \approx \prod_i P(w_i | w_{i-k}, \cdots, w_{i-1})$$

  - And we have:

$$P(w_i | w_1, w_2, \cdots, w_{i-1}) \approx P(w_i | w_{i-k}, \cdots, w_{i-1})$$

Andrei Markov

- Simplifying Language Model
  - Bigram (N-Gram, N = 2)

$$P(learn | Never, too, late, to) \approx P(learn | to)$$

  - Trigram (N-Gram, N = 3)

$$P(learn | Never, too, late, to) \approx P(learn | late, to)$$

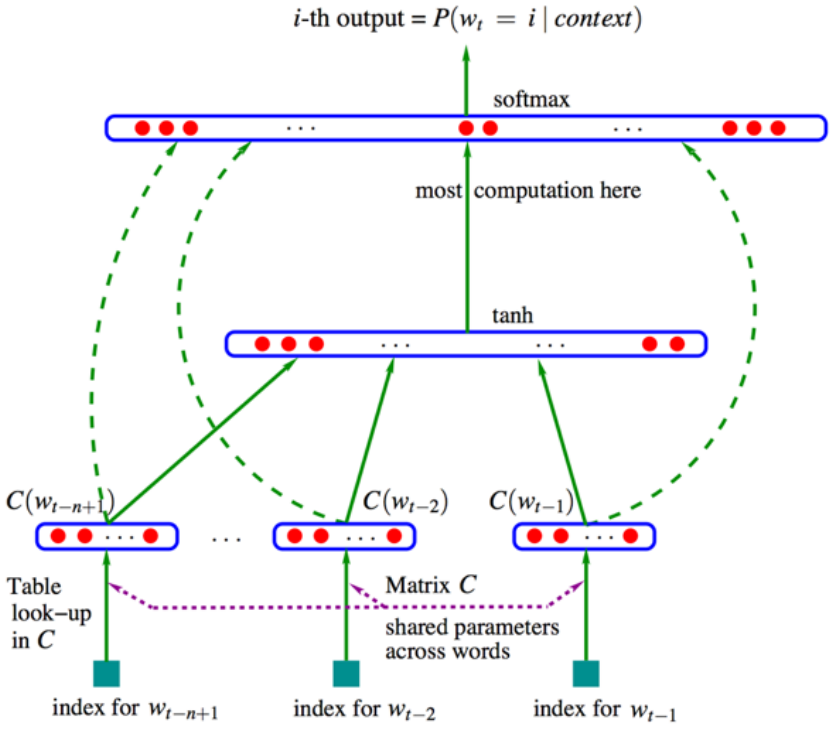
Problems of N-gram

- Not considering contexts farther than 1 or 2 words
- Not capturing the similarity between words
  - The cat is walking in the bedroom
  - A dog was running in a room
    - N-gram cannot find the similar semantics and grammatical role between cat and dog, walking and running

Neural Language Model

- A neural language model is a language model based on neural networks to learn distributed representations of words
  - Associate words with distributed vectors
  - Compute the joint probability of word sequences in terms of the feature vectors
  - Optimize the word feature vectors (embedding matrix E) and the parameters of the loss function (map matrix W)

Neural Language Model



Journal of Machine Learning Research 3 (2003) 1137–1155 Submitted 4/02; Published 2/03

A Neural Probabilistic Language Model

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Réjean Ducharme DUCHARME@IRO.UMONTREAL.CA
Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Christian Jauvin JAUVINC@IRO.UMONTREAL.CA
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal, Montréal, Québec, Canada

Editors: Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

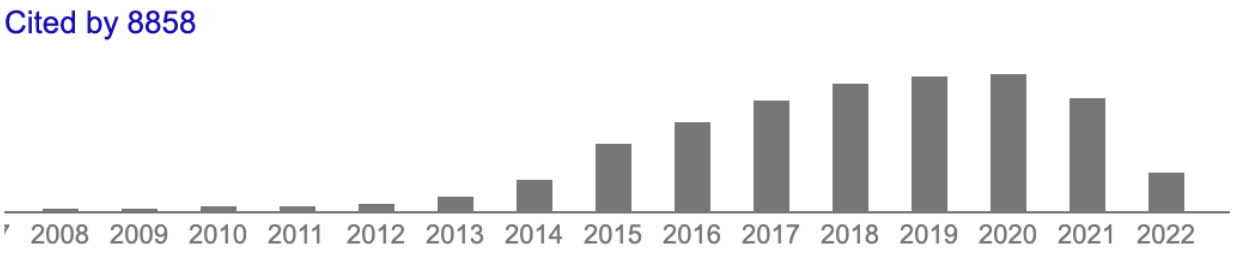
Abstract

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models (with millions of parameters) within a reasonable time is itself a significant challenge. We report on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-the-art n-gram models, and that the proposed approach allows to take advantage of longer contexts.

Keywords: Statistical language modeling, artificial neural networks, distributed representation, curse of dimensionality

1. Introduction

A fundamental problem that makes language modeling and other learning problems difficult is the *curse of dimensionality*. It is particularly obvious in the case when one wants to model the joint distribution between many discrete random variables (such as words in a sentence, or discrete attributes in a data-mining task). For example, if one wants to model the joint distribution of 10 consecutive words in a natural language with a vocabulary V of size 100,000, there are potentially $100000^{10} - 1 = 10^{50} - 1$ free parameters. When modeling continuous variables, we obtain generalization more easily (e.g. with smooth classes of functions like multi-layer neural networks or Gaussian mixture models) because the function to be learned can be expected to have some local smoothness properties. For discrete spaces, the generalization structure is not as obvious: any change of these discrete variables may have a drastic impact on the value of the function to be esti-



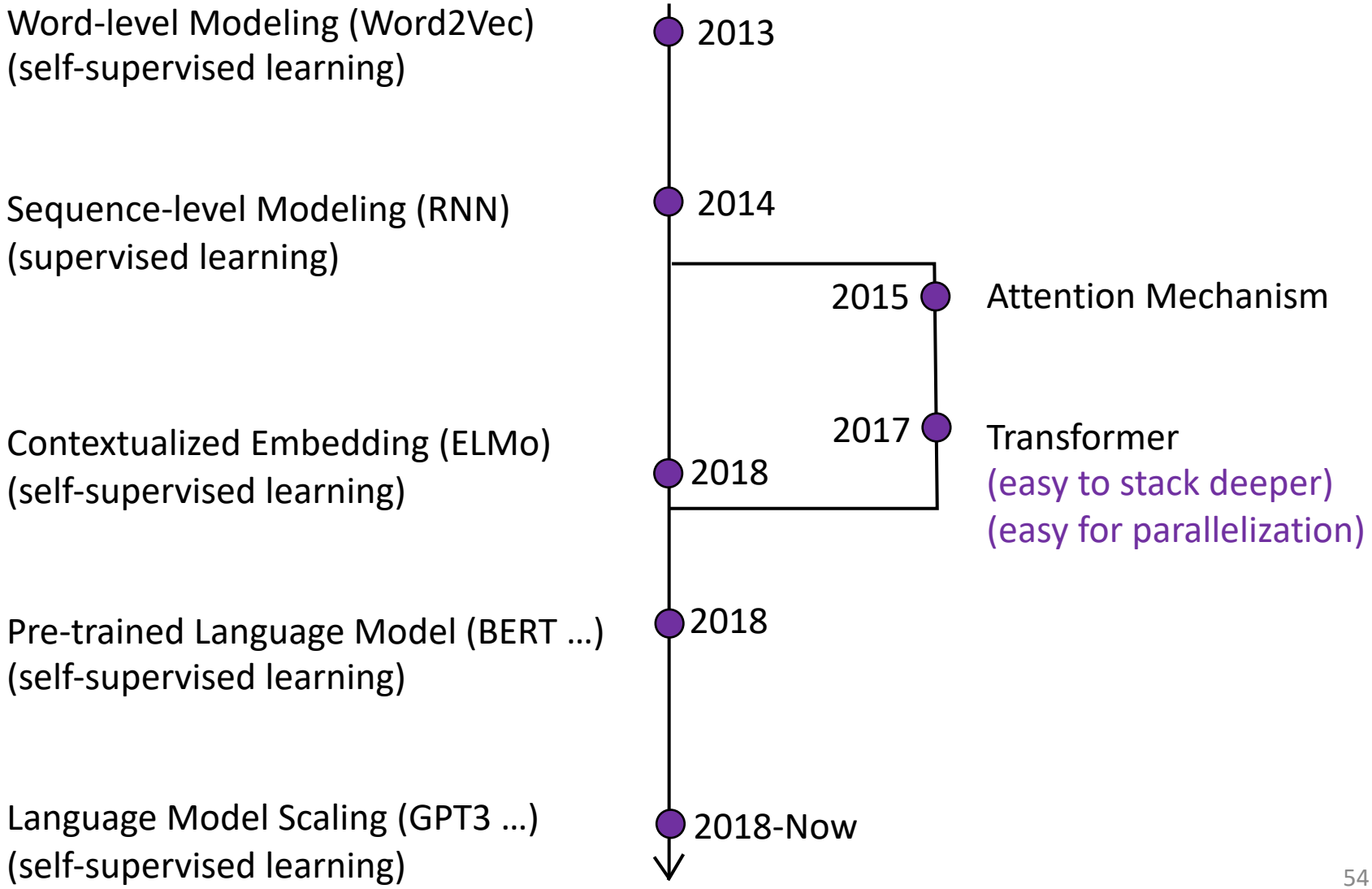
Yoshua Bengio, et al. A neural probabilistic language model. JMLR 2003.

Big Model Basics
The Trip to Big Models

Xu Han

THUNLP

The Trip to Big PLMs

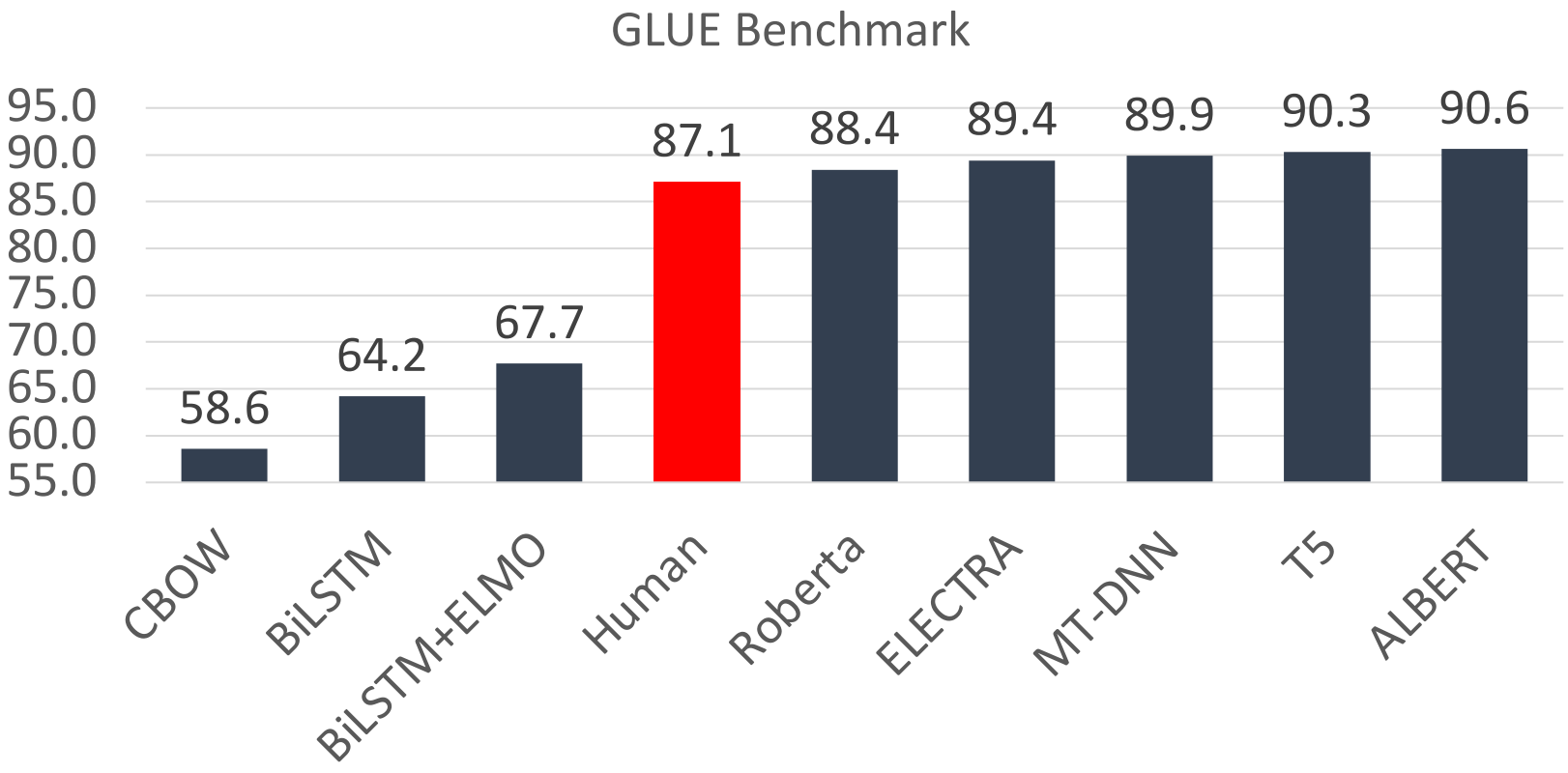


Big Model Basics
Why are Big Models Important

THUNLP

Why PLMs

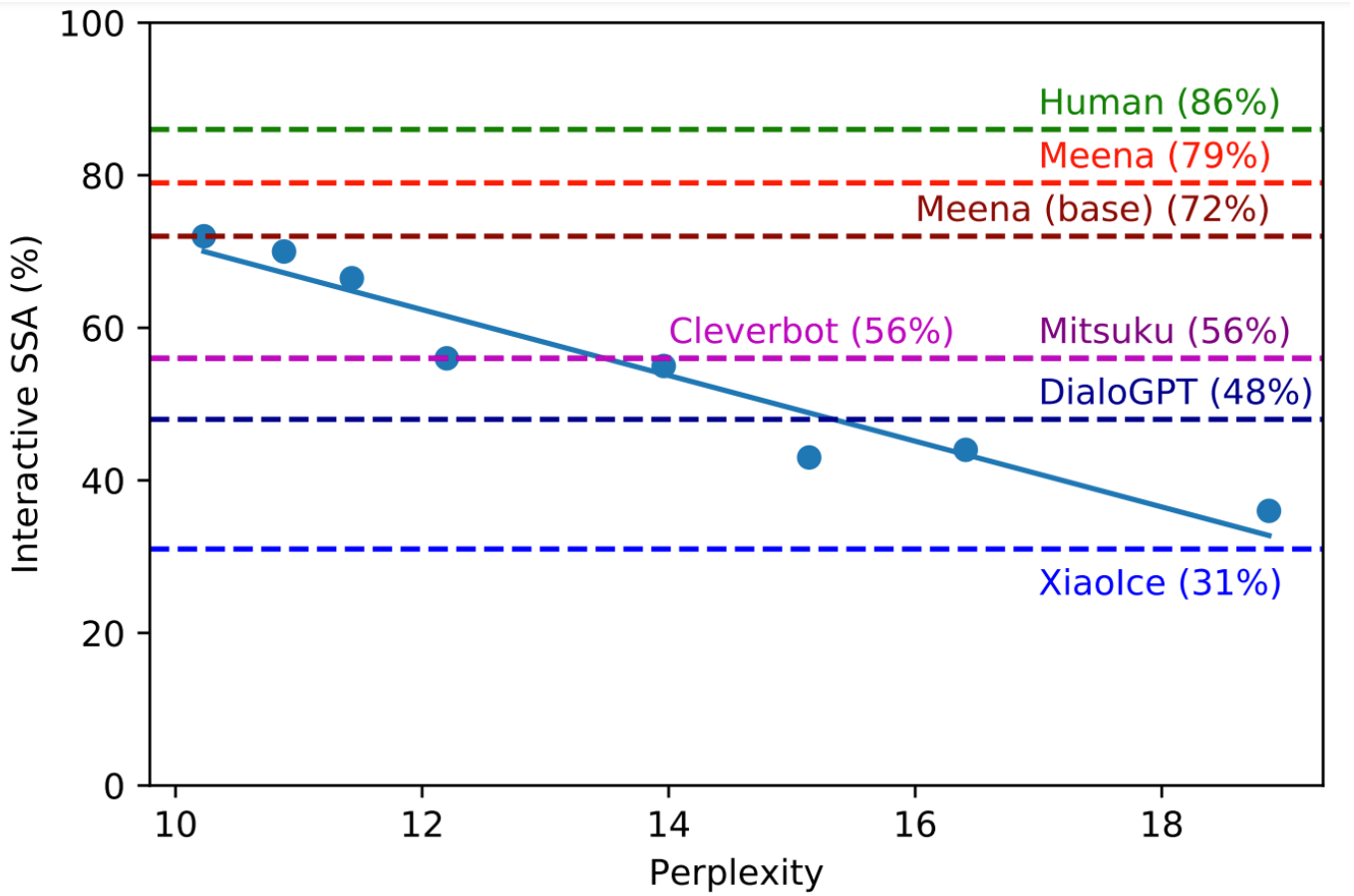
The results of the pre-trained language model on GLUE surpass human level, reflecting the ability of language understanding



https://gluebenchmark.com/

Why PLMs

The pre-trained language model has also made breakthroughs in the dialogue system, reflecting the ability of language generation

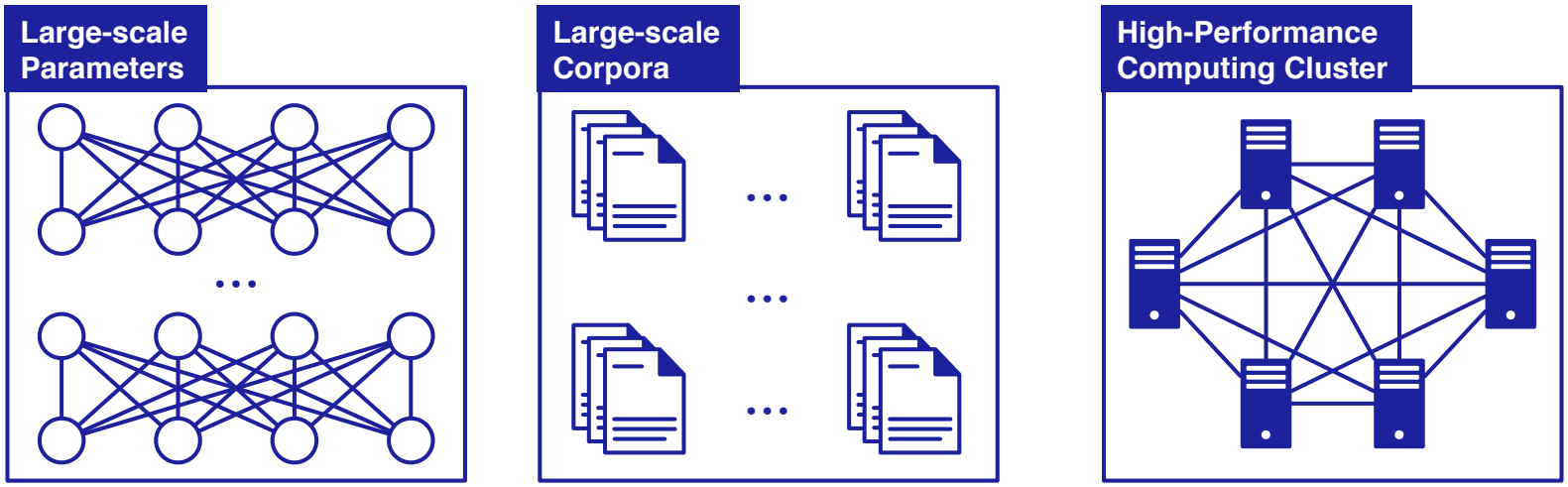


https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html

Why Big Models

After 2018, three characteristics of pre-trained language models: Large amount of parameters, large amount of data, large amount of computation

Increasing parameters and data is an effective means to further improve performance



Why Big Models

The scale of pre-trained language models has grown at a rate of about 10 times per year in the past two years

The amount of data is also growing accordingly, and the computational cost is becoming more and more expensive

Time	Org	Model	Model Scale	Data Scale	Training Time
2018.6	OpenAI	GPT	110M	4GB	3d
2018.10	Google	BERT	330M	16GB	50d
2019.2	OpenAI	GPT-2	1.5B	40GB	200d
2019.7	Facebook	RoBERTa	330M	160GB	3y
2019.10	Google	T5	11B	800GB	66y
2020.6	OpenAI	GPT-3	175B	560GB	90y

Note: M-millions, B-billions. The last column training time is the estimated time for training with a single NVIDIA V100 GPU

Why Big Models

GPT-3 has knowledge and can reason to some extent

World Knowledge

Q: Who was president of the United States in 1801?
A: Thomas Jefferson was president of the United States in 1801.

Q: What city is in the northwest corner of Ohio?
A: Toledo is in the northwest corner of Ohio.

Q: Who won the World Series in 2021?
A: The New York Yankees won the World Series in 2021.

Q: How many Super Bowls do the Cincinnati Bengals win in the 2030s?
A: The Cincinnati Bengals win two Super Bowls in the 2030s.

Common Sense

Q: How many eyes does a giraffe have?
A: A giraffe has two eyes.

Q: How many legs does a frog have?
A: A frog has four legs.

Q: Are there any animals with three legs?
A: No, there are no animals with three legs.

Q: Why don't animals have three legs?
A: Animals don't have three legs because they would fall over.

Logical Reasoning

Q: If I have a marble and a paper clip in a box, put a pencil in the box, and remove the marble, what is left?
A: A paper clip.

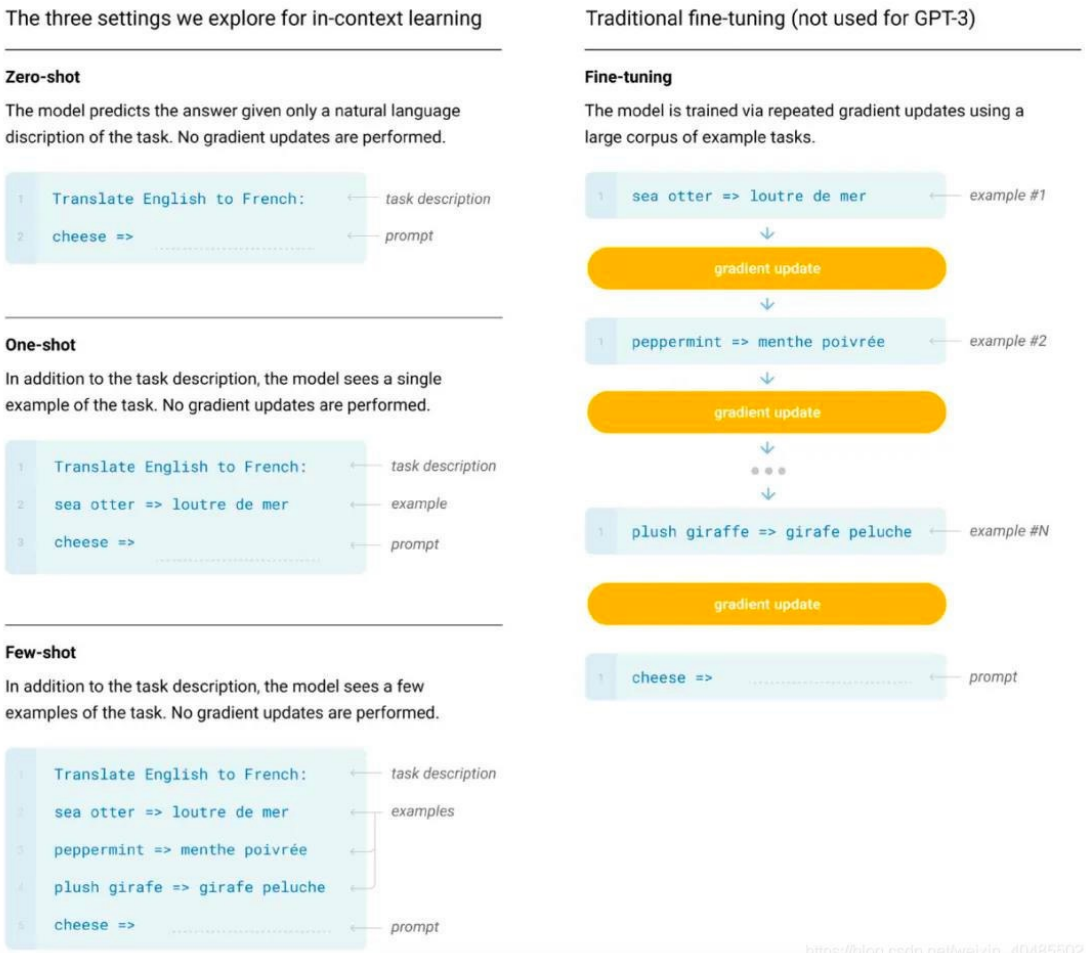
Q: If I have two shoes in a box, put a pencil in the box, and remove one shoe, what is left?
A: A shoe.

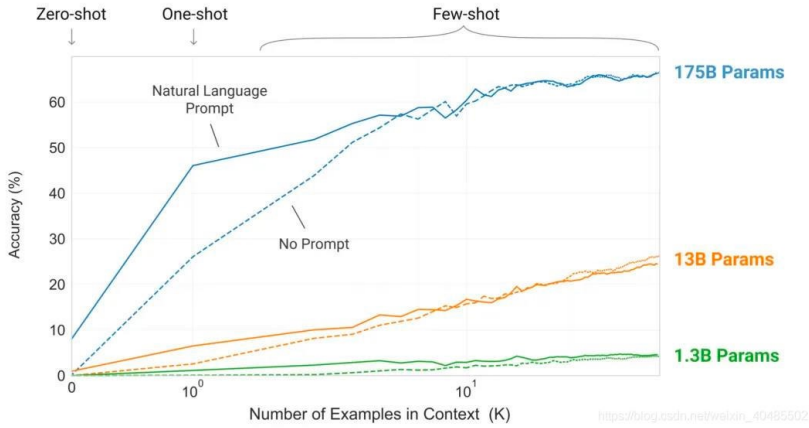
Q: If I put a pencil in a box, then put another pencil in the box, what is in the box?
A: Two pencils.

Q: Then if I take out a pencil and put in a frog, what is in the box?
A: A frog.

Why Big Models

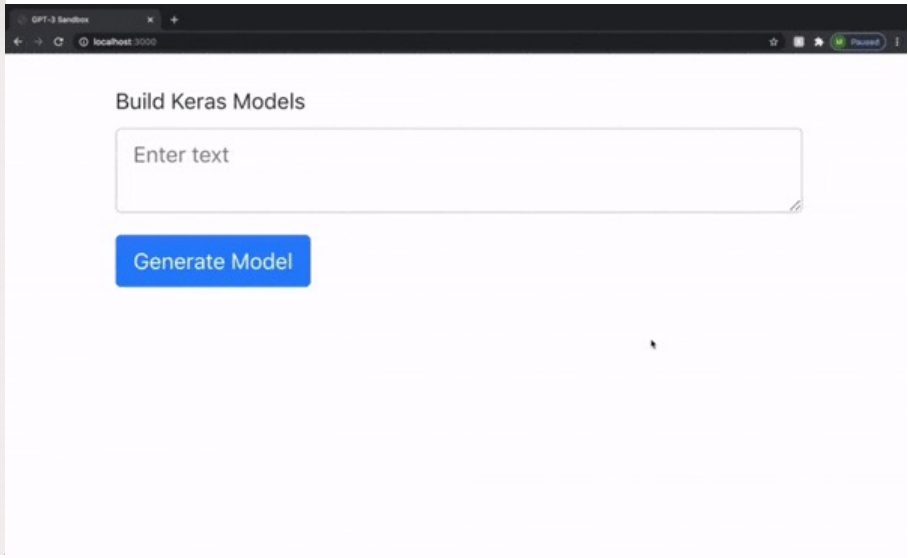
In addition, GPT-3 has a powerful zero/few-shot learning ability to perform many tasks





Why Big Models

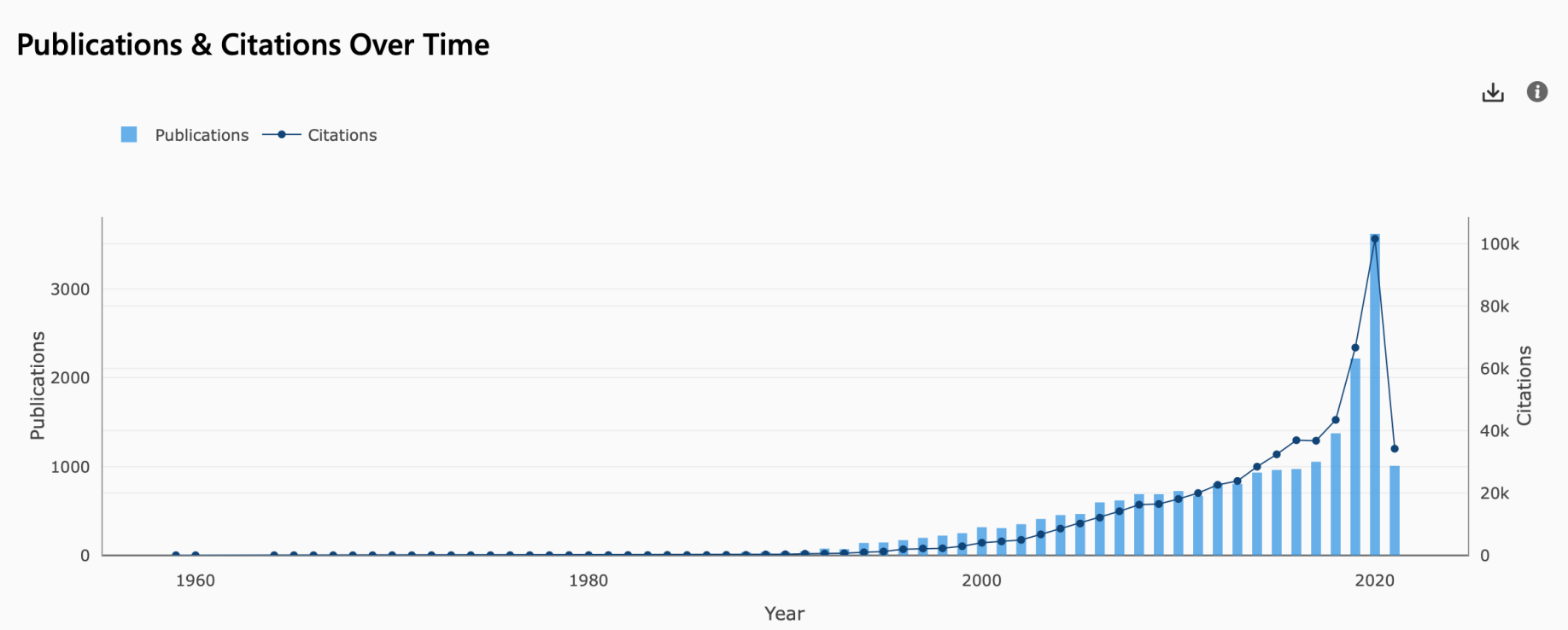
In addition, GPT-3 has a powerful zero/few-shot learning ability to perform many tasks



Why Big Models

The use of pre-trained language models is now standard for various NLP tasks

Language model related research has also grown rapidly after 2018



https://academic.microsoft.com/topic/137293760/publication/

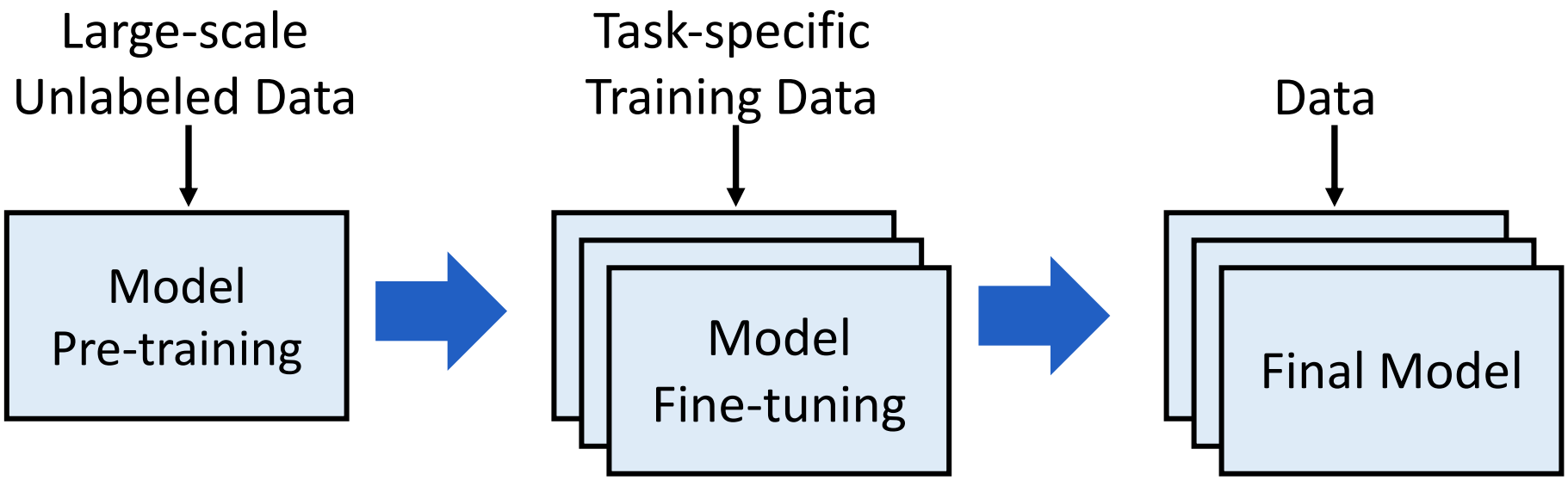
Big Model Basics
Paradigms Behind Big Models

THUNLP

Paradigms behind Big Models

In the pre-training stage, pre-trained language models **capture rich knowledge from large-scale unlabeled data**

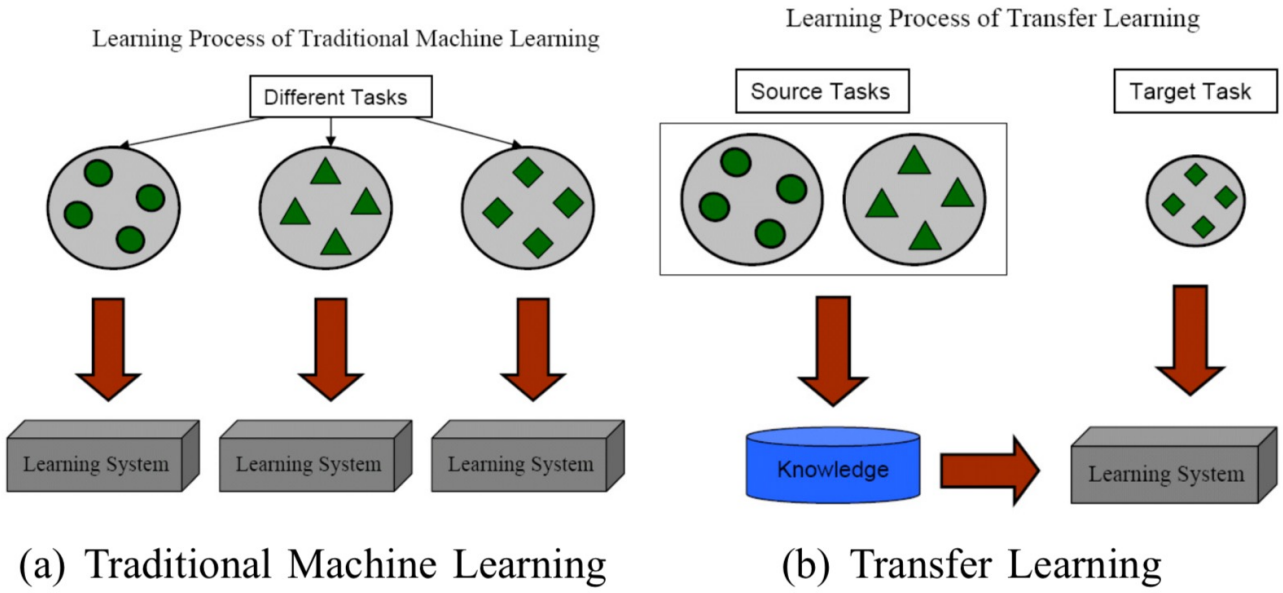
Then, we can fine-tune pre-trained language models with task-specific training data **to adapt the pre-trained knowledge**



Paradigms behind Big Models

The basic paradigm of pre-training and fine-tuning can be traced back to transfer learning

Humans can apply previously learned knowledge to handle new problems faster, and we want machines to have similar abilities



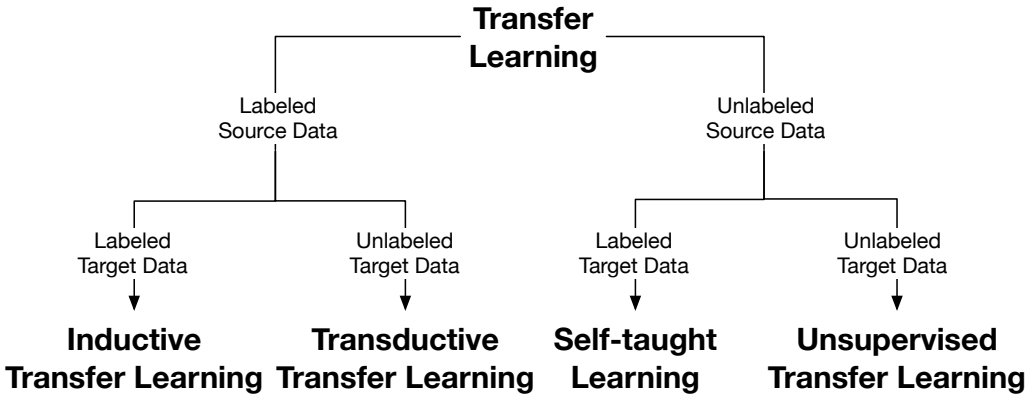
(a) Traditional Machine Learning (b) Transfer Learning

A survey on transfer learning.

Paradigms behind Big Models

Transfer learning uses a "pre-training and then fine-tuning" framework to achieve "knowledge acquisition and then knowledge transfer"

Both feature-representation-transfer and parameter-transfer are used in the subsequent works of pre-training models



	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
Instance-transfer	√	√	
Feature-representation-transfer	√	√	√
Parameter-transfer	√		
Relational-knowledge-transfer	√		

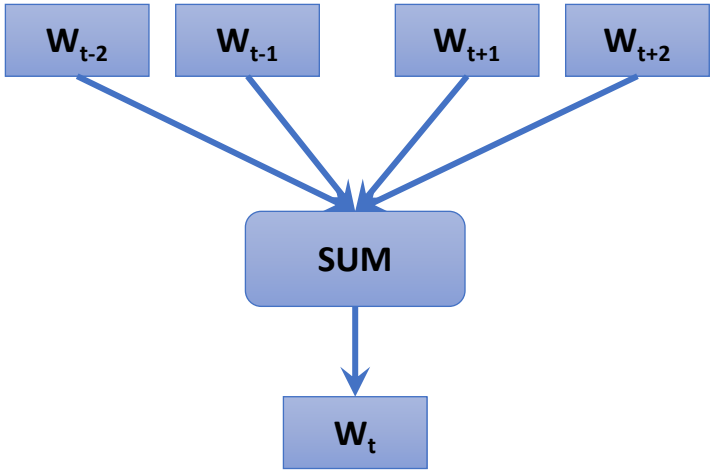
A survey on transfer learning.

Paradigms behind Big Models

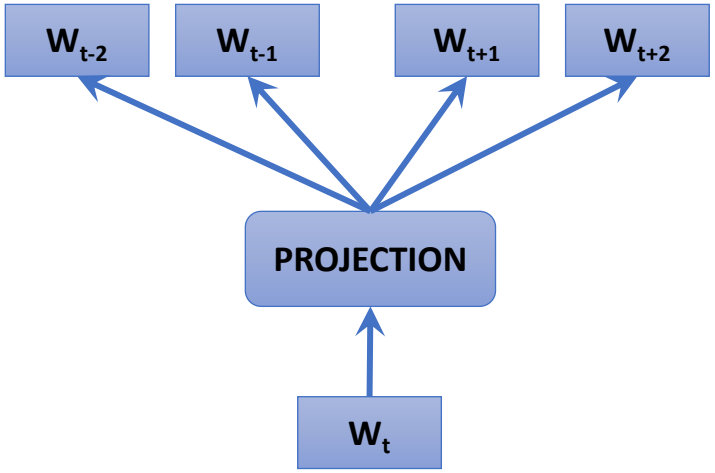
Word embeddings can be seen as simple pre-trained language models, such as CBOW and Skip-Gram

The word embeddings learned from unlabeled data in a self-supervised manner can be used as input for downstream tasks

This is feature-representation-transfer



CBOW



Skip-Gram

Efficient Estimation of Word Representations in Vector Space

Paradigms behind Big Models

The problems of word embeddings

Hard to detect polysemy

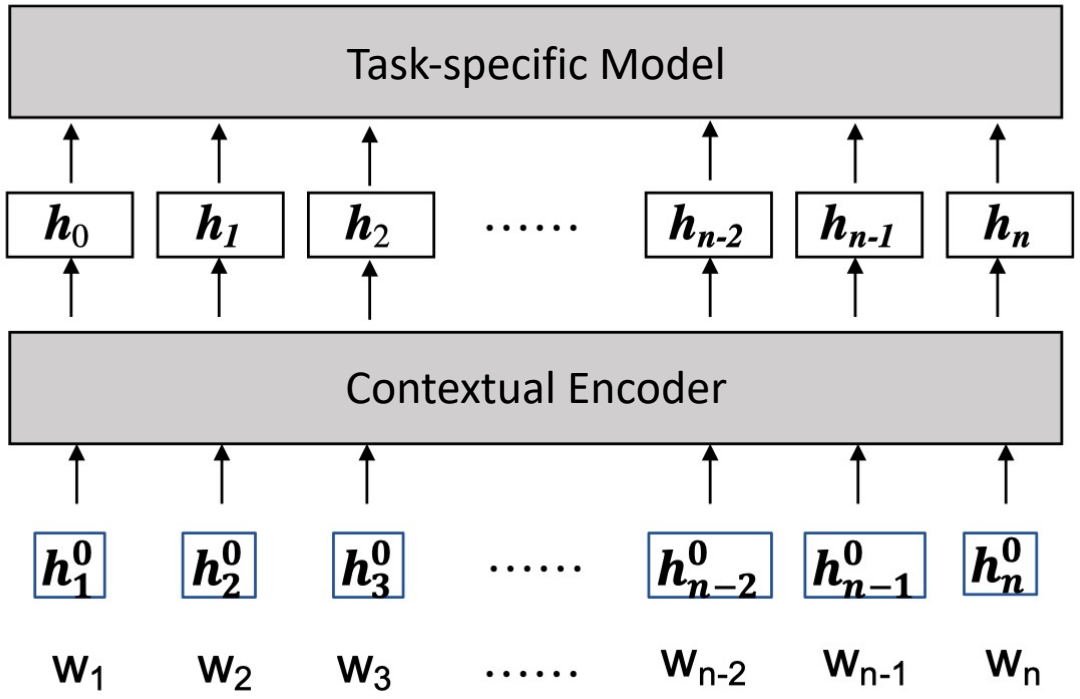
- I go to bank for money deposit
- I go to bank for fishing

Hard to detect antonym

- I love this movie. The movie is so bad
- I don’t love this movie. The movie is so good

Paradigms behind Big Models

Solution: contextual word embeddings

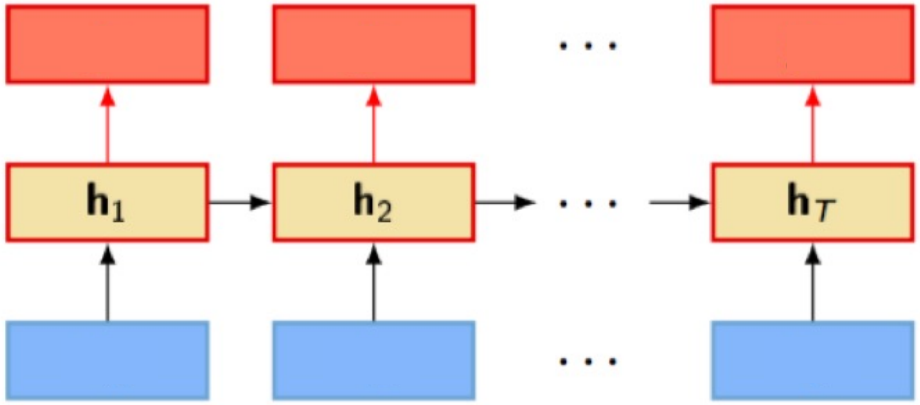


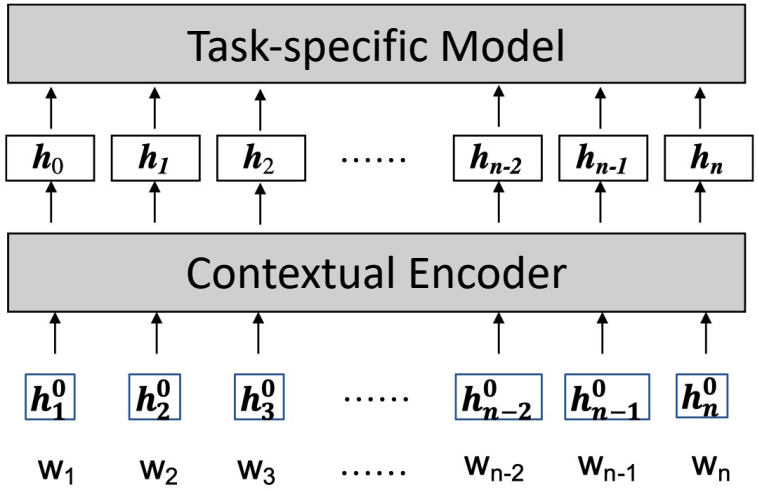
Paradigms behind Big Models

Solution: contextual word embeddings

ELMo:

- Perform language modeling on large-scale unlabeled data using RNNs
- Use the pre-trained RNNs to generate contextual word embeddings for task-specific model





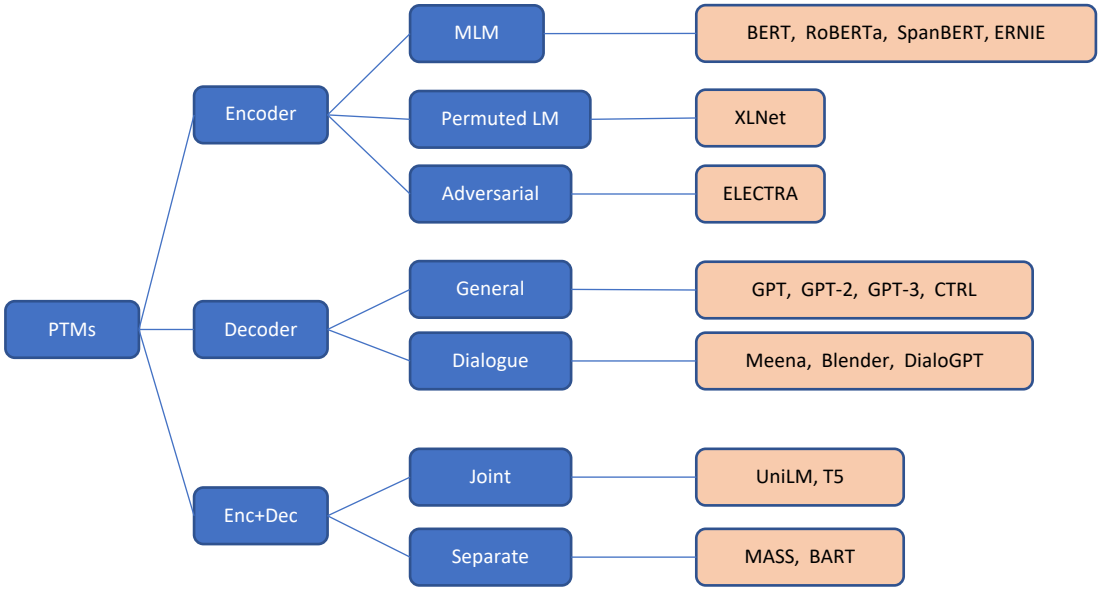
ELMo: Deep contextualized word representations

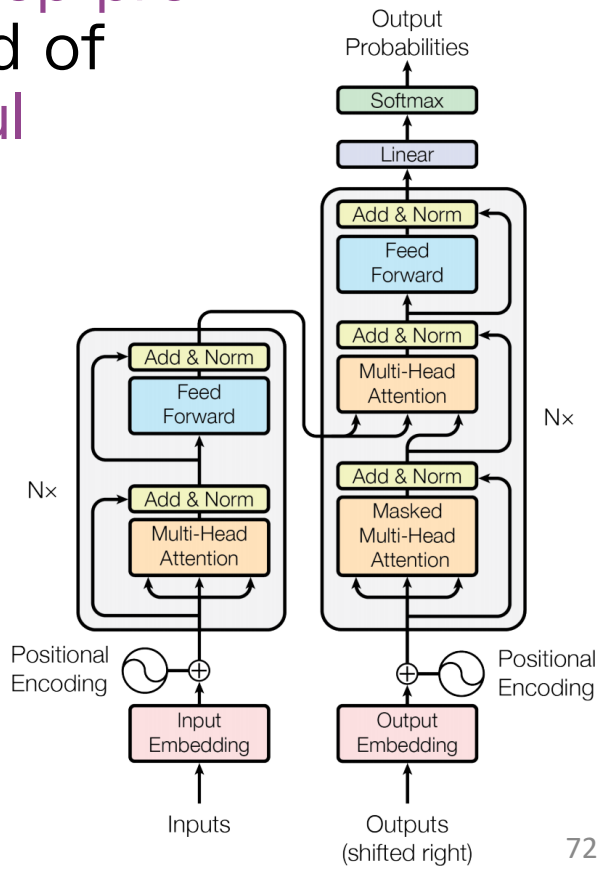
ULMFiT: Universal Language Model Fine-tuning for Text Classification

Paradigms behind Big Models

The breakthrough of NLP: Transformer

Based on Transformer, a series of deep pre-training models are developed instead of shallow RNNs, which is more powerful





Attention is All You Need.

Big Model Basics
Typical Case of Big Models

THUNLP

Typical Case of Big Models

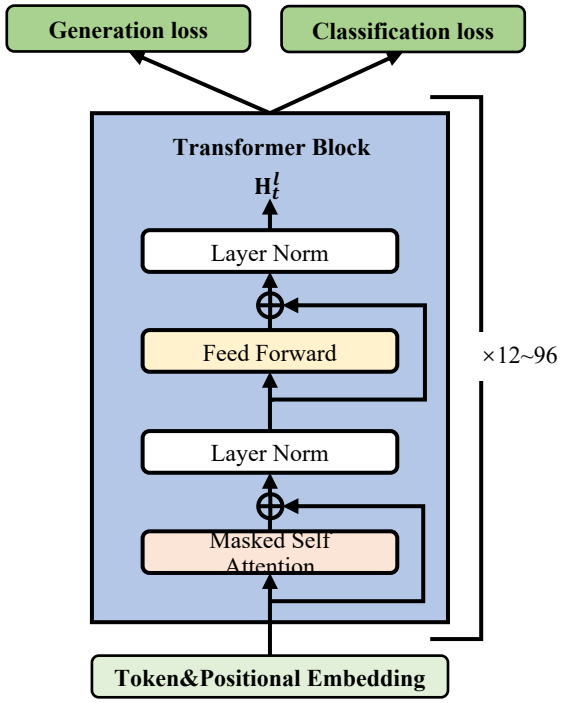
GPT:

perform language modeling as a pre-training task

$$L_1(C) = -\sum_i \log P(w_i | w_1 w_2 \dots w_{i-1})$$

can be used in downstream generation tasks (as decoder), language understanding tasks (as encoder)

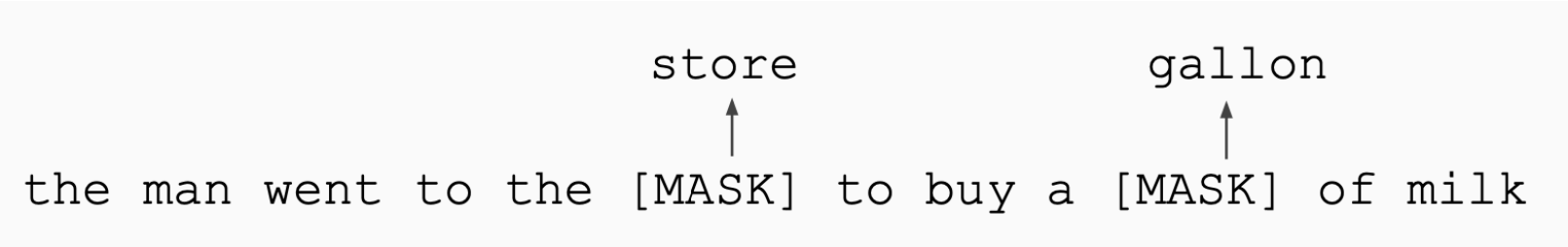
$$L_2(C) = -\sum_{(u,y)} \log P(y | w_1 w_2 \dots w_T)$$



Typical Case of Big Models

BERT:

Masked language model (MLM)



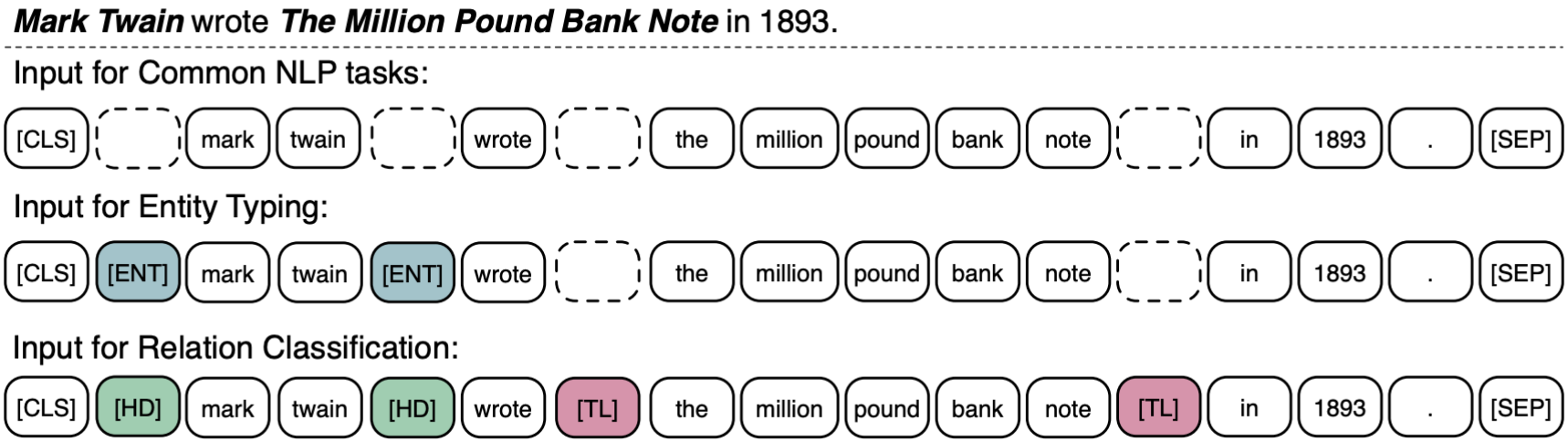
Next sentence prediction (NSP, discourse-level)

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Typical Case of Big Models

- BERT:
- Add specific symbols for fine-tuning of downstream tasks

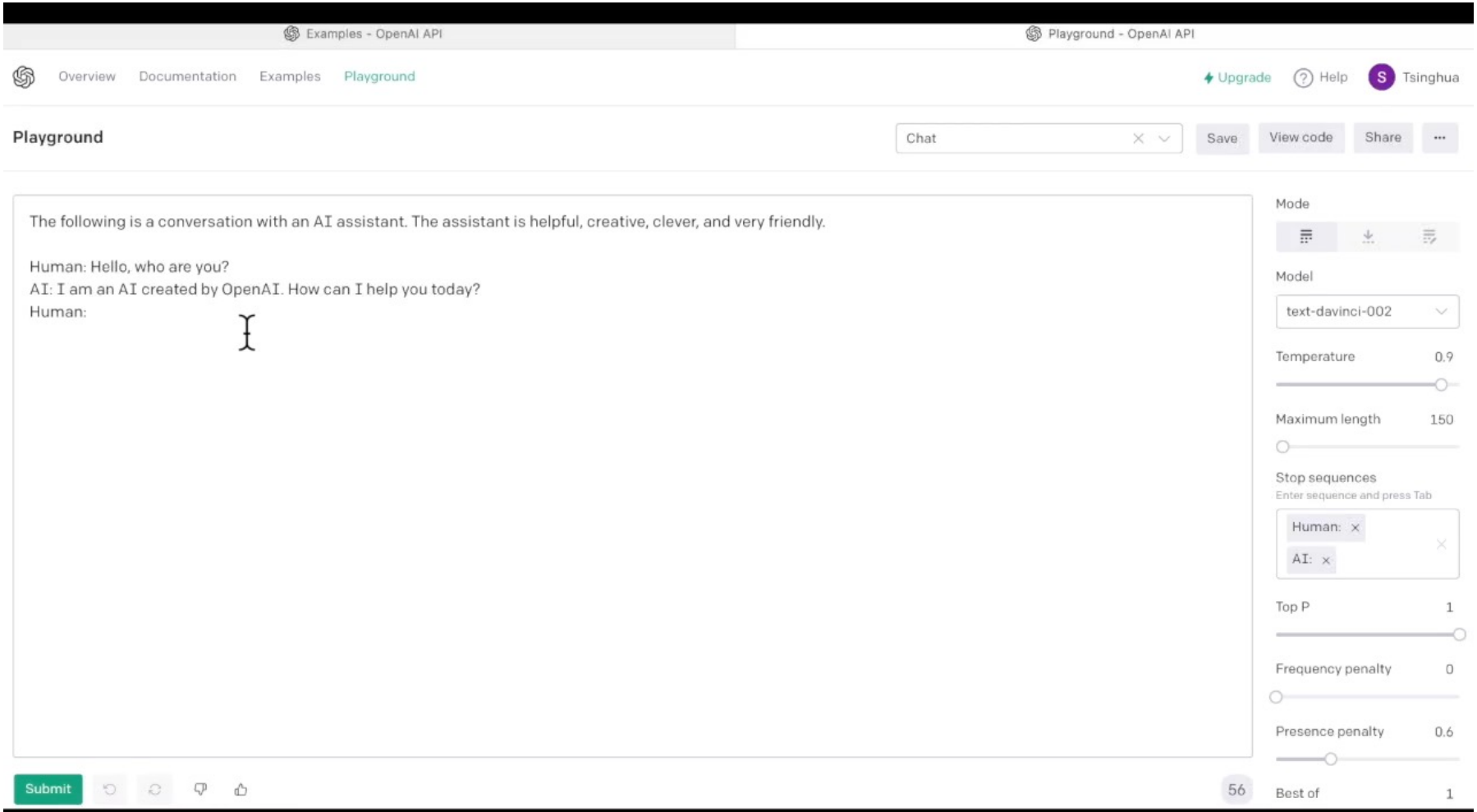


Big Model Basics Demos

THUNLP

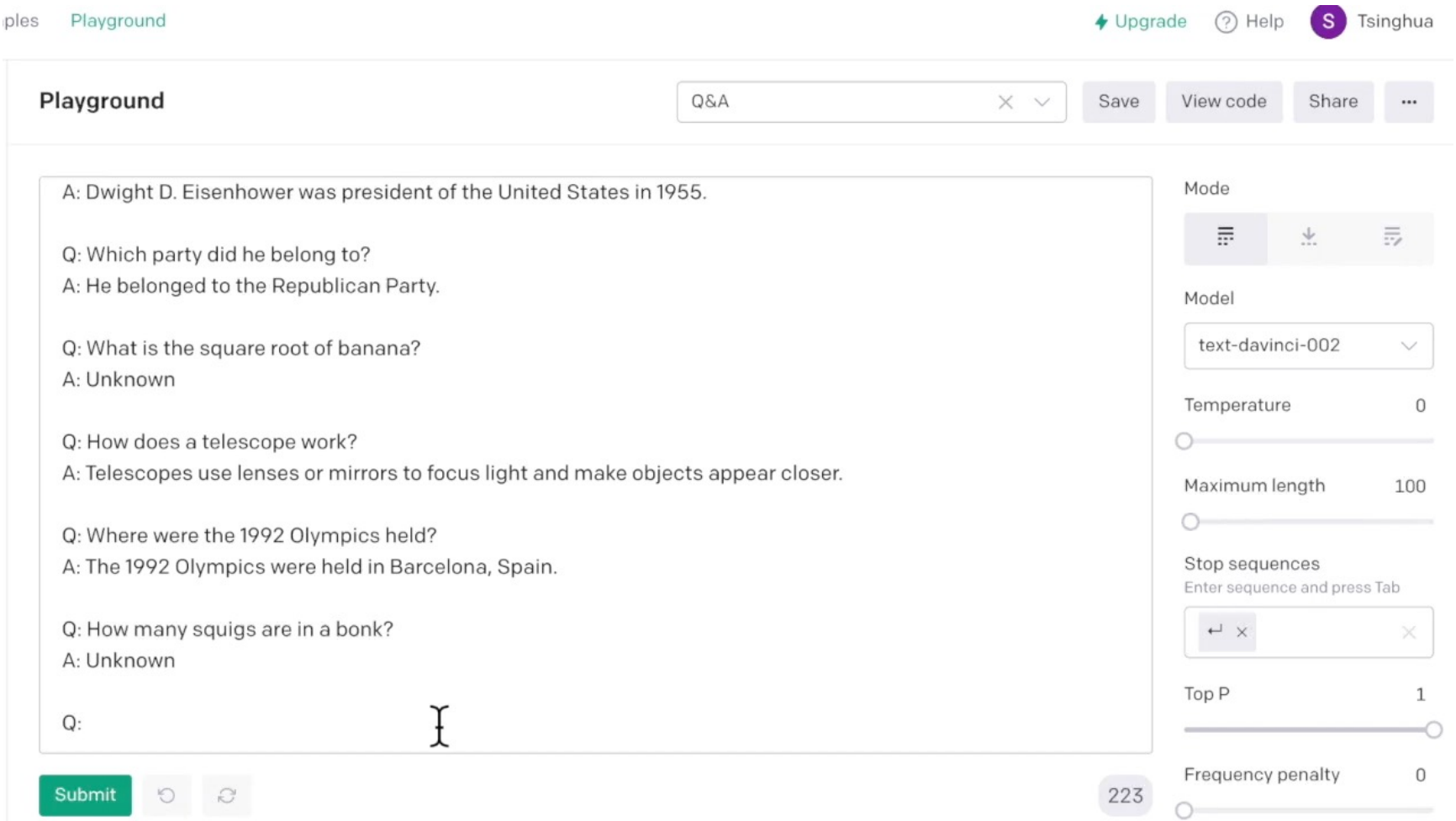
Demos of Big Models

- Human-level chatting with GPT-3 (175B)



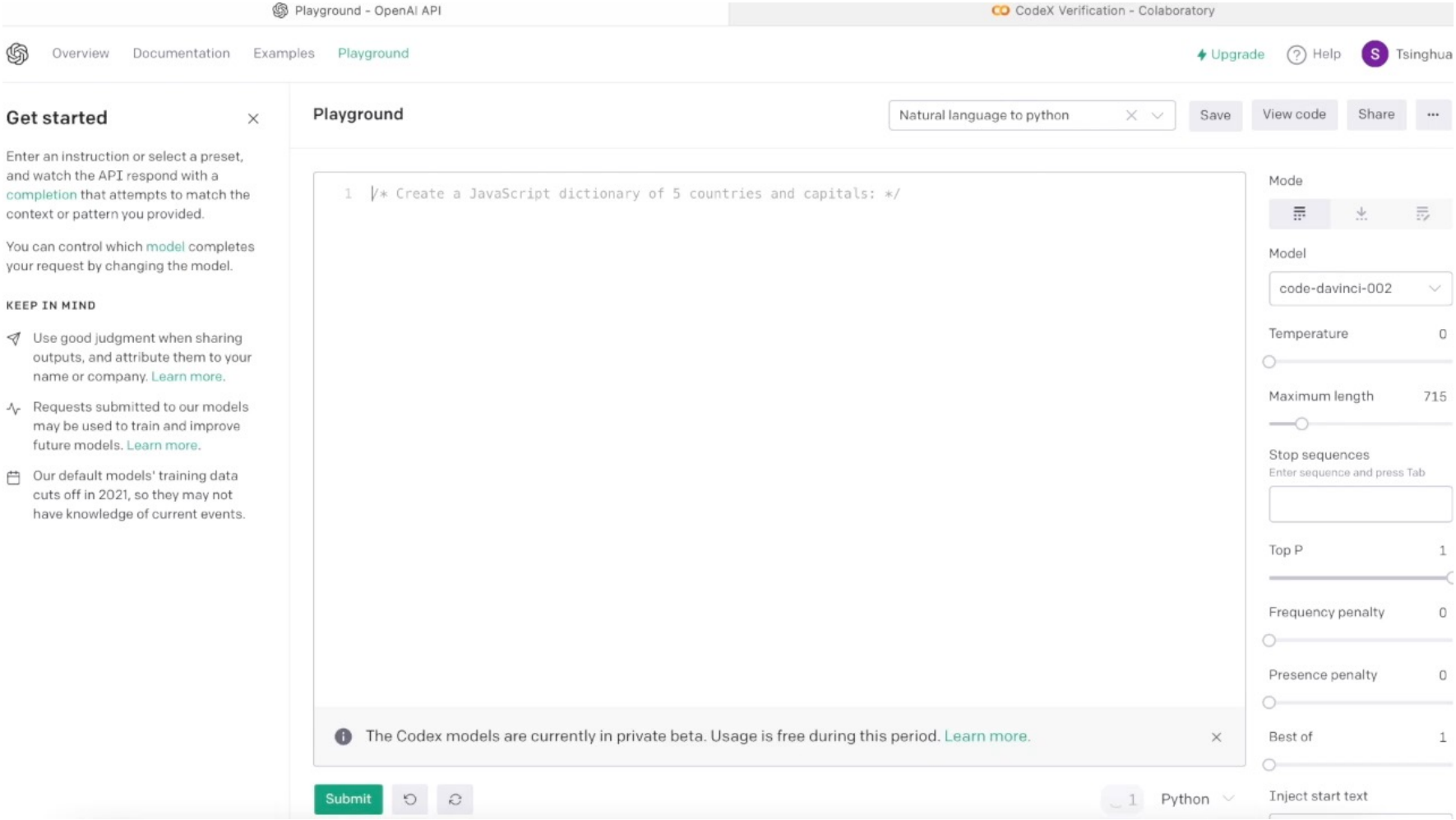
Demos of Big Models

- Knowledgeable question answering with GPT-3 (175B)



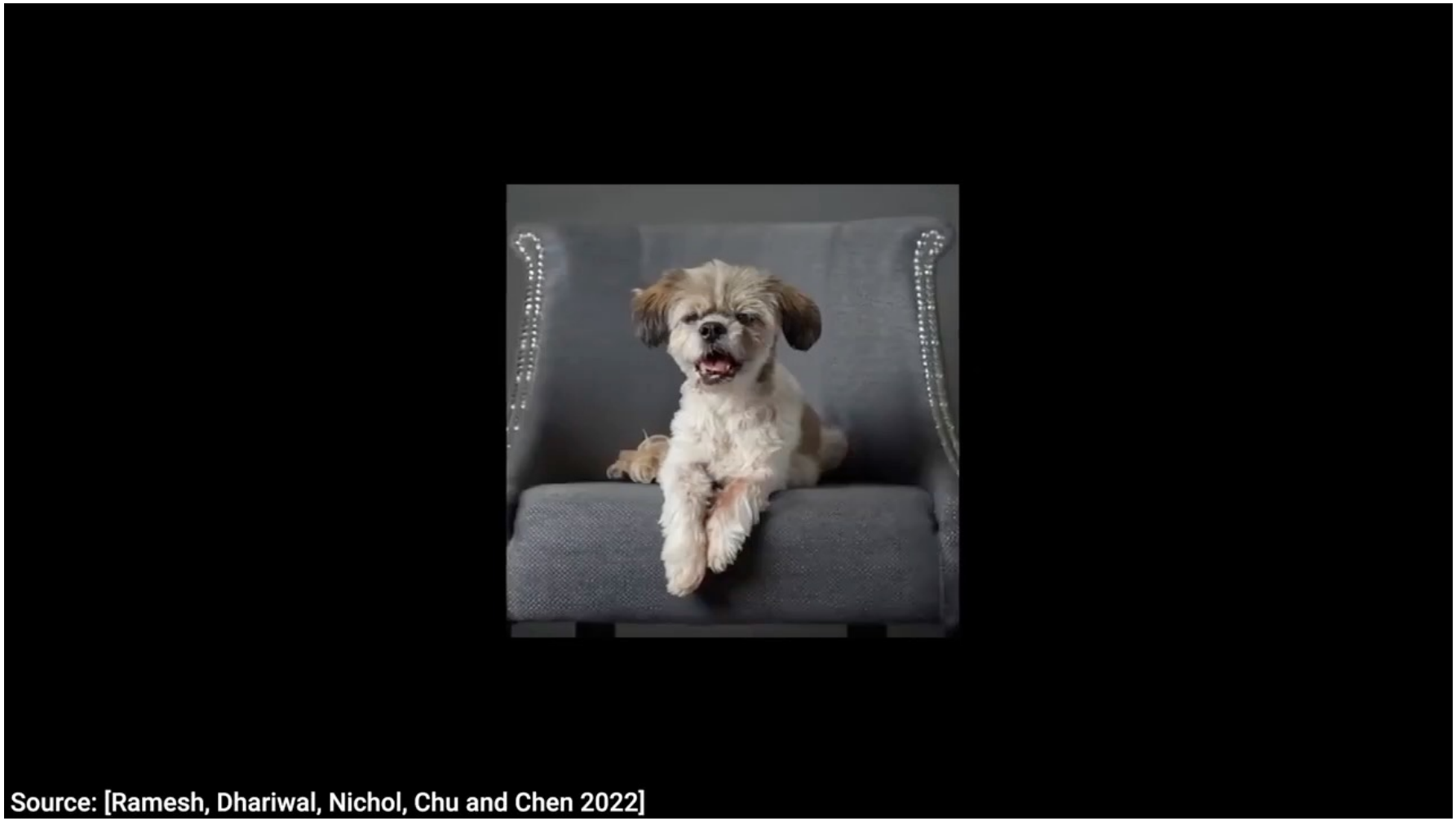
Demos of Big Models

- Scientific plotting with CodeX (175B)



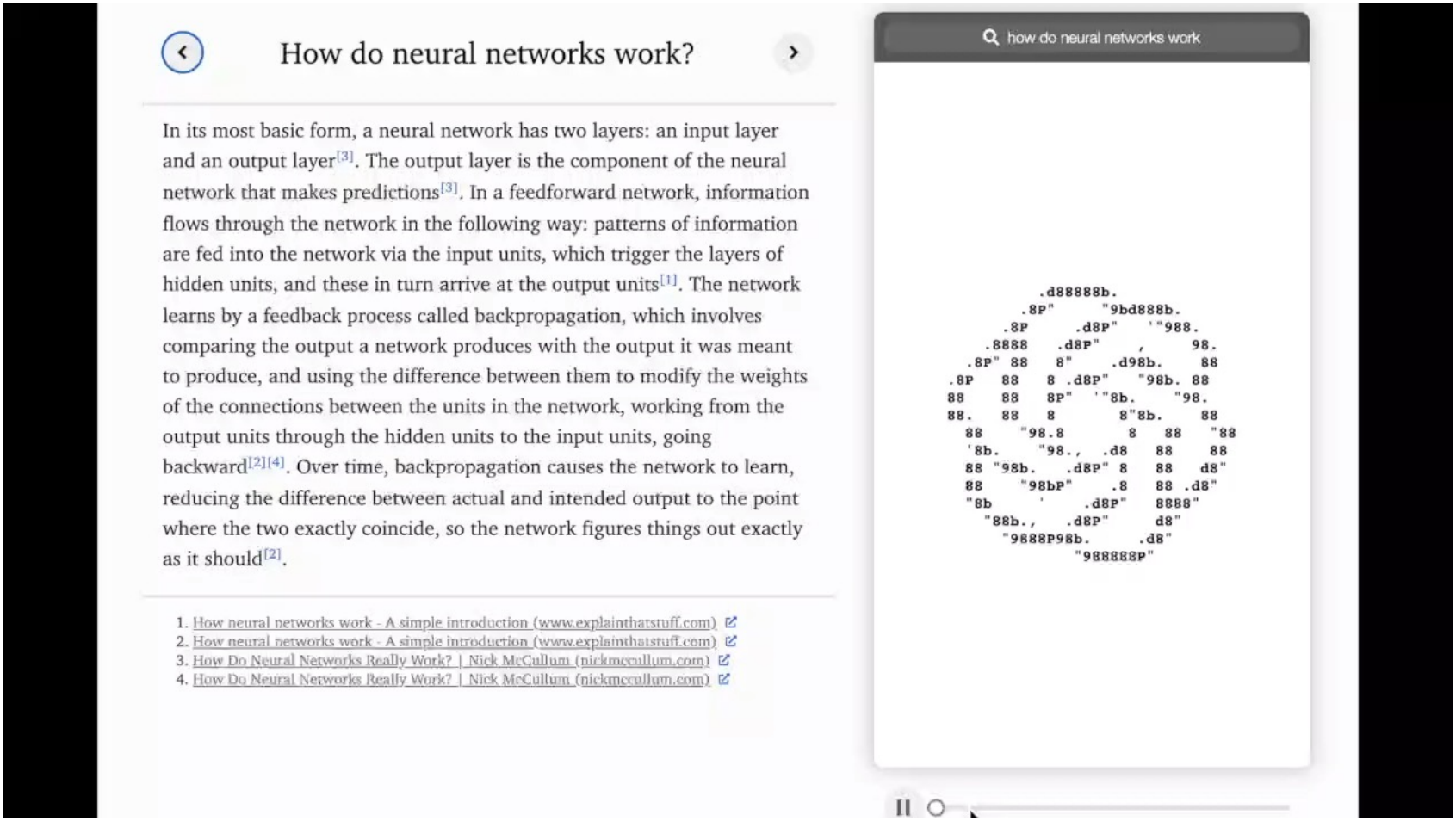
Demos of Big Models

- Image generation with DALL-E 2 (5B)



Demos of Big Models

- Web-based question answering with WebGPT



Coding Environment & GPU Server

Shengding Hu

THUNLP

Server & GPU

- Every registered student will get an account on ParaCloud
- **Nvidia RTX3090, about 200 Hours**

- Notice: DO NOT use it for personal use.

https://www.paratera.com/

Server & GPU

- For unregistered students, you can
  - Try to get GPU server by yourself
  - Use google colab (sometimes can have GPU/TPU resources allocated)
    - Unstable
    - Buy a pro account to become stable.

https://colab.research.google.com/

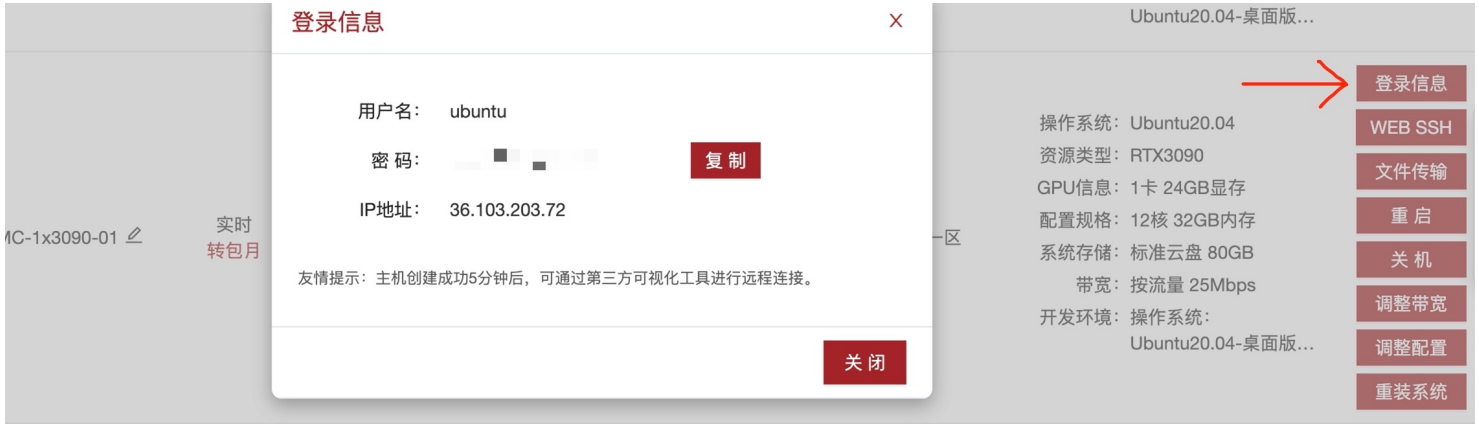
Prerequisites

- SSH
- Linux command
- Vim
- Tmux
- Virtual environment & conda & pip
- Vscode + remote connection
- Git
- Bash

Search engines always help!

ssh login

Follow the instruction on web learning to login to the server



ssh login

Follow the instruction on web learning to login to the server

```
00:48:55 ✘ hsd@Shengding-MBP ~ ⬡ v16.14.2 2h9m4s
$ ssh ubuntu@36.103.203.72
The authenticity of host '36.103.203.72 (36.103.203.72)' can't be established.
ED25519 key fingerprint is SHA256:QEFKgfD2fh7XXVEqTheiPed6sBtHcRzd/fLht7TVk8k.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:47: 36.103.203.91
    ~/.ssh/known_hosts:53: 36.103.203.87
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '36.103.203.72' (ED25519) to the list of known hosts.
ubuntu@36.103.203.72's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-104-generic x86_64)
```

Useful Linux Commands

- **mkdir xxx**: create a folder with the name xxx
- **ls xxx**: list all files in the folder xxx. Use **ls** **or** **ls ./** to quickly show all files in the current folder
- **cd xxx**: change the current working directory to xxx. Use **cd ../** to change to parent directory of the current working directory
- **vim xxx**: open text editor for xxx file in terminal

A Simple Example

- Let‘s say you want to **create a folder** named *nlp*, **write a code** to print hello world, and **save the code** in the course folder.

- **mkdir xxx**: create a folder with the name xxx
- **ls xxx**: list all files in the folder xxx. Use **ls** **or** **ls ./** to quickly show all files in the current folder

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~$ mkdir nlp
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos  anaconda3  nlp
```

A Simple Example

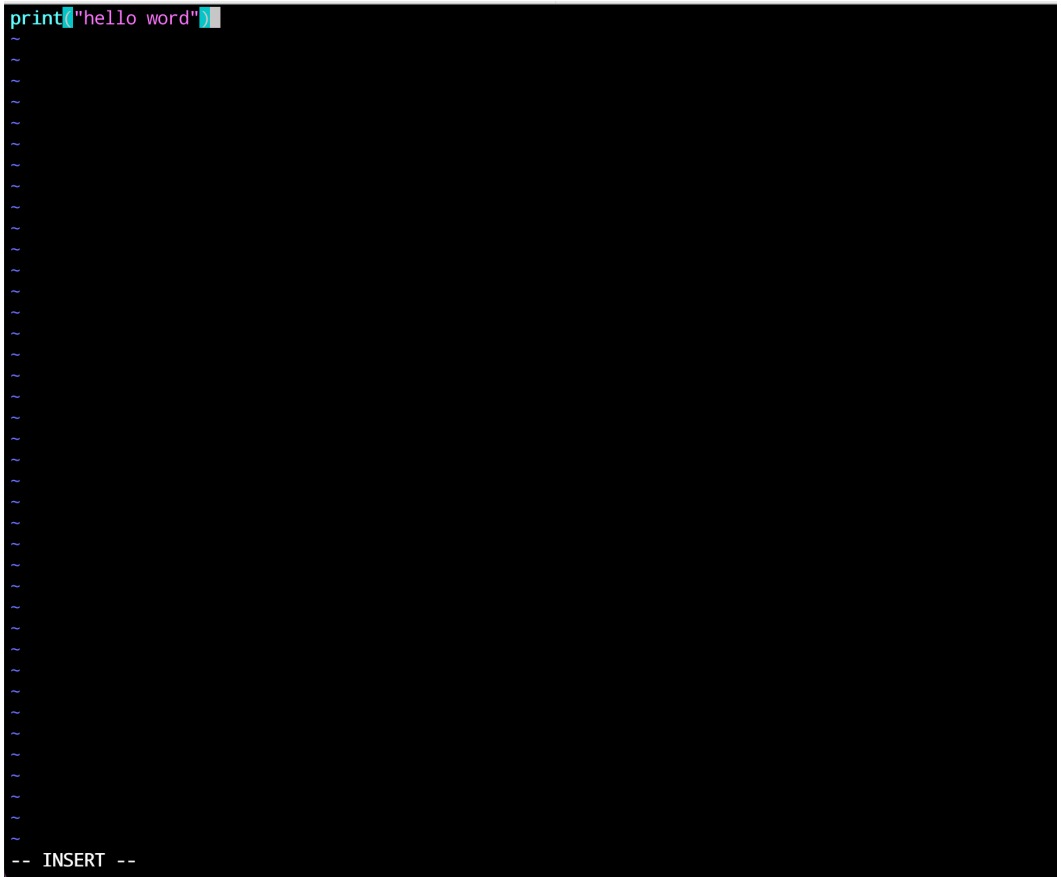
- Change the working directory into the course folder *nlp*
- Use vim to create a file named *my_code.py, type* vim *my_code.py*

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~$ cd nlp
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ vim my_code.py
```

we are now inside the `nlp` folder

A Simple Example

- You will open an empty file. Press i to enter insert mode write the code, press **ESC** key, type :wq to save and exit the editor.



A Simple Example

- Actually, using vim is not simple…

How do I exit the Vim editor?

Asked 9 years, 6 months ago Active 1 month ago Viewed 2.6m times

4575

I'm stuck and cannot escape. It says:

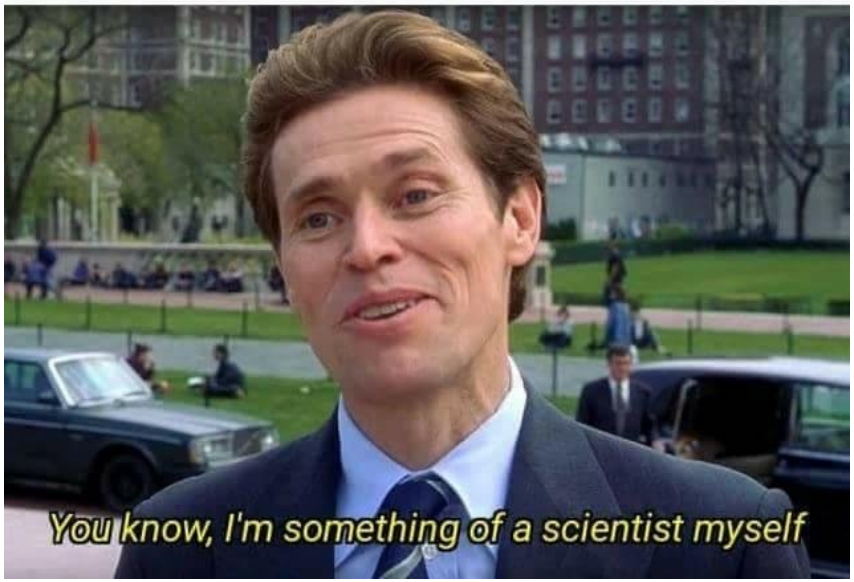
```
"type :quit<Enter> to quit VIM"
```

But when I type that it simply appears in the object body.

1049

vim vi

When you finally exit vim



A Simple Example

- *my_code.py* is now in the folder.
- Type python3 my_code.py to execute the file

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ python3 my_code.py
hello word
```

Tmux to mange sessions

- We suggest you use tmux to manage sessions

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~$ tmux new -s first_session
```

- The session will keep alive when you log out.

https://tmuxcheatsheet.com/

Use Virtual Environment

- Check that conda is installed

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ conda
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications,

Options:

positional arguments:
  command
    clean        Remove unused packages and caches.
    compare      Compare packages between conda environme
    config       Modify configuration values in .condarc.
                 (/home/ubuntu/.condarc) by default.
```

Use Virtual Environment

- Check nvcc version (for later install the correct pytorch version)
- From below, we have 11.5, keep it in mind when installing pytorch

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Thu_Nov_18_09:45:30_PST_2021
Cuda compilation tools, release 11.5, V11.5.119
Build cuda_11.5.r11.5/compiler.30672275_0
```

https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html

Use Virtual Environment

- Create an virtual environment with anaconda:

```
conda create -n [envname] python=3.8
```

-n : environment name

python=3.8: specify the version of python to be 3.8 in the environment

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ conda create -n py38general python=3.8
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.10.3
  latest version: 4.13.0

Please update conda by running

    $ conda update -n base -c defaults conda
```

Use Virtual Environment

- Activate the created environment: `conda activate [envname]` You will see a prefix appears on the command line

```
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ conda activate py38general
(py38general) ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ python
Python 3.8.13 (default, Mar 28 2022, 11:38:47)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use Virtual Environment

- Install packages
- In your activated virtual environment
- Conda install
- conda install pytorch torchvision torchaudio cudatoolkit=11.1 -c pytorch-lts -c nvidia
- Pip install

```
pip install numpy
```

Use Virtual Environment

https://download.pytorch.org/whl/torch_stable.html

- Directly download the wheel for better version control
- wget https://download.pytorch.org/whl/cu115/torch-1.11.0%2Bcu115-cp38-cp38-linux_x86_64.whl

```
pip install torch-1.11.0%2Bcu115-cp38-cp38m-linux_x86_64.whl
```

Use Virtual Environment

- Verify that PyTorch is installed correctly by importing torch in python interpreter.

```
(py38general) ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~$ python
Python 3.8.13 (default, Mar 28 2022, 11:38:47)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>>
```

Use GPU on ParaCloud

- Write a simple GPU code snippet as *check_gpu.py:*

check_gpu.py

```
import torch
print(torch.cuda.is_available())
```

- Write a script to run the code as *run.sh*

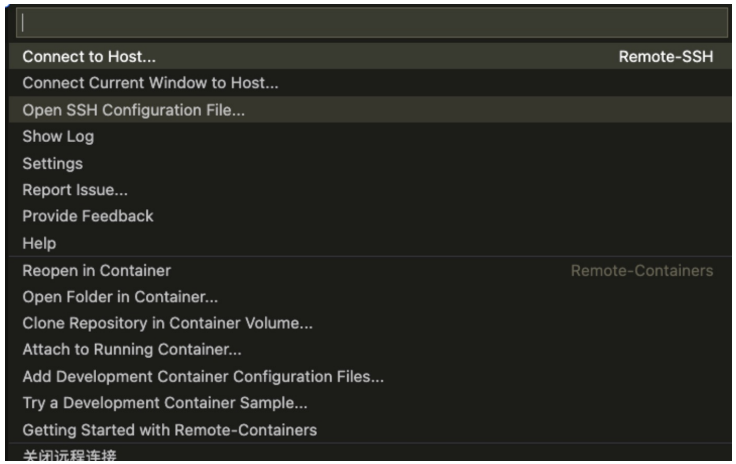
run.sh

```
#!/bin/bash
python check_gpu.py
```

Vscode – remote connection

- vscode: we suggest use vscode as the main IDE
- Use vscode to connect to the server
  - Click on the >< button

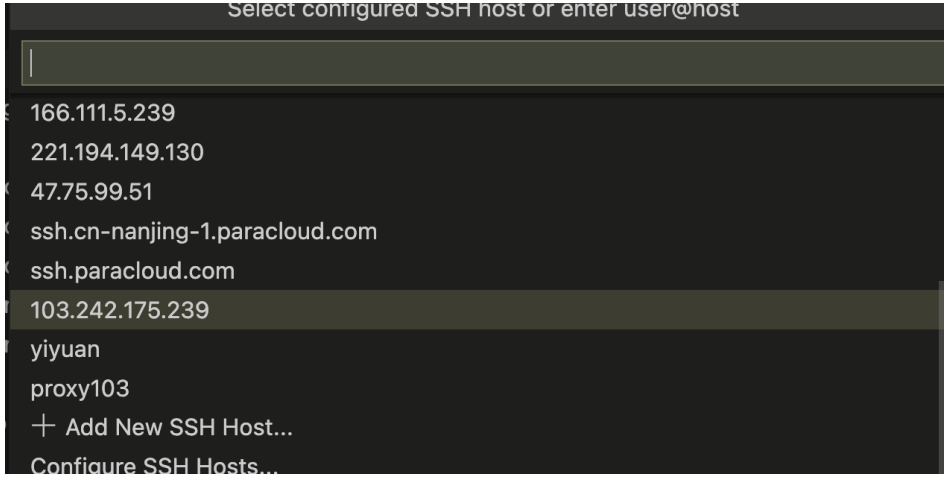
  - Select open ssh configuration file, add connection info



Vscode – remote connection

- Click on the >< button again

- Select connect to host,
- Select the server that you just entered, enter password,
- Now you log in to the server, select your folder to enter



Now we try a pytorch script

- https://pytorch.org/tutorials/beginner/examples_nn/polynomial_module.html#sphx-glr-beginner-examples-nn-polynomial-module-py

src/pytorch_nn.py

```python
import torch
import math


class Polynomial3(torch.nn.Module):
    def __init__(self):
        """
        In the constructor we instantiate four parameters and assign them as
        member parameters.
        """
        super().__init__()
        self.a = torch.nn.Parameter(torch.randn(()))
        self.b = torch.nn.Parameter(torch.randn(()))
        self.c = torch.nn.Parameter(torch.randn(()))
        self.d = torch.nn.Parameter(torch.randn(()))

    def forward(self, x):
        """
        In the forward function we accept a Tensor of input data and we must return
        a Tensor of output data. We can use Modules defined in the constructor as
        well as arbitrary operators on Tensors.
        """
        return self.a + self.b * x + self.c * x ** 2 + self.d * x ** 3

    def string(self):
        """
        Just like any class in Python, you can also define custom method on PyTorch modules
        """
        return f'y = {self.a.item()} + {self.b.item()} x + {self.c.item()} x^2 + {self.d.item()}

# Create Tensors to hold input and outputs.
x = torch.linspace(-math.pi, math.pi, 2000)
y = torch.sin(x)

# Construct our model by instantiating the class defined above
model = Polynomial3()

# Construct our loss function and an Optimizer. The call to model.parameters()
# in the SGD constructor will contain the learnable parameters (defined
# with torch.nn.Parameter) which are members of the model.
criterion = torch.nn.MSELoss(reduction='sum')
optimizer = torch.optim.SGD(model.parameters(), lr=1e-6)
for t in range(2000):
    # Forward pass: Compute predicted y by passing x to the model
    y_pred = model(x)

    # Compute and print loss
    loss = criterion(y_pred, y)
    if t % 100 == 99:
        print(t, loss.item())

    # Zero gradients, perform a backward pass, and update the weights.
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

print(f'Result: {model.string()}')
```

scripts/pytorch_nn.sh

```
scripts > pytorch_nn.sh
1  #!/bin/sh
2
3  python src/pytorch_nn.py
4
```

bash scripts/pytorch_nn.sh

Now we try a pytorch script

```
(py38general) ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ bash scripts/pytorch_nn.sh
99 1301.2197265625
199 866.116455078125
299 577.6156616210938
399 386.2862548828125
499 259.37469482421875
599 175.17489624023438
699 119.30001831054688
799 82.2126693725586
899 57.58972930908203
999 41.23762130737305
1099 30.37517738342285
1199 23.157371520996094
1299 18.359783172607422
1399 15.169775009155273
1499 13.047927856445312
1599 11.636051177978516
1699 10.696237564086914
1799 10.0703706741333
1899 9.653388977050781
1999 9.375459671020508
Result: y = 0.009554248303174973 + 0.8355141878128052 x + -0.0016482671489939094 x^2 + -0.09031108021736145 x^3
```

GitHub repo

- Run ssh-keygen on your machine

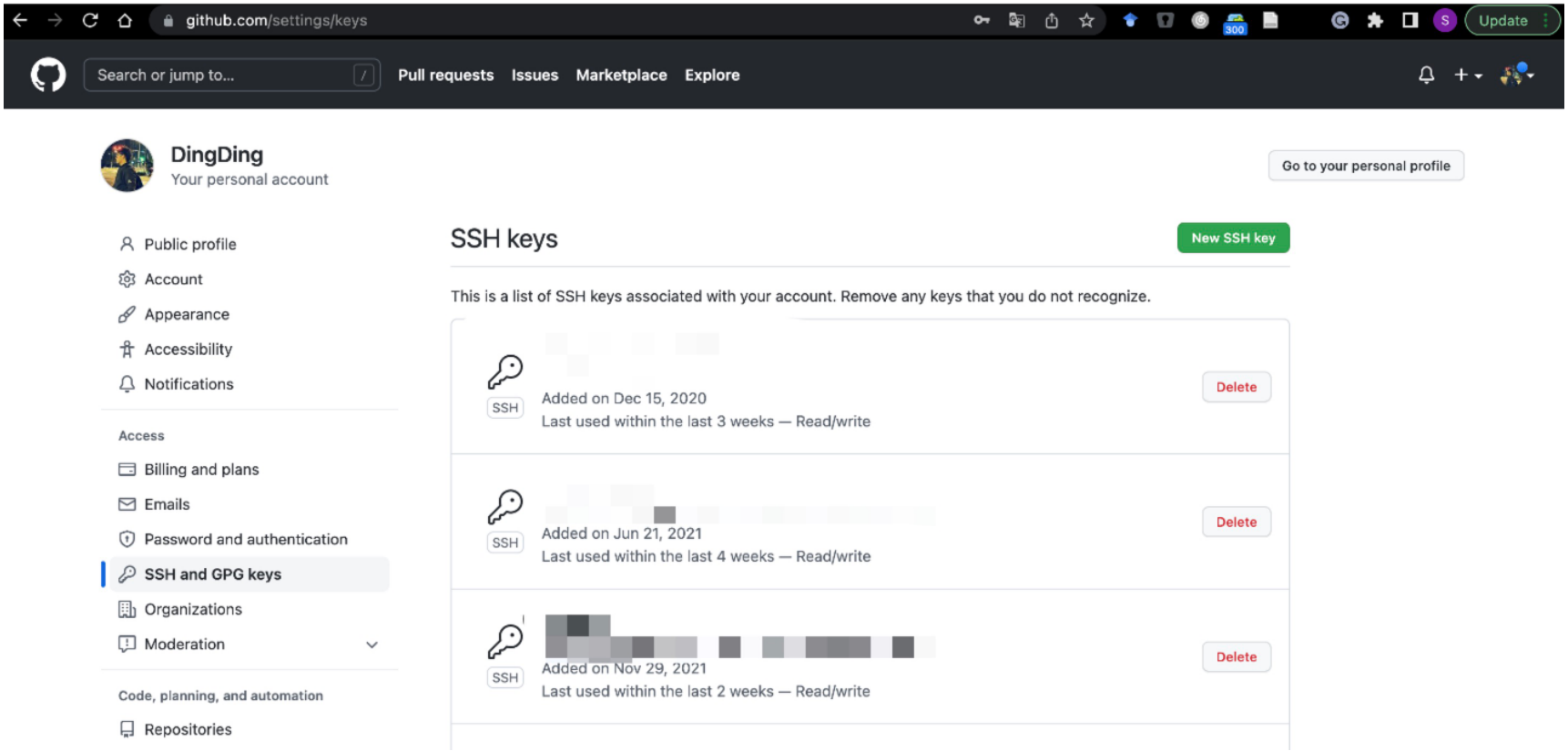
```
(py38general) ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:6P1c1tTxweKvZLMsg0FPZmIVPxfkt0dIL3DmfKlauhs ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m
The key's randomart image is:
+---[RSA 3072]----+
|           .o =o |
|           ..Booo|
|          .  +=B=|
|       . + +. ==*|
|      . S *  +..+|
|     . . . .+o. .|
|      . . oEo+.. |
|         + =* +  |
|          oo++   |
+----[SHA256]-----+
```

cat ~/.ssh/id_rsa.pub

```
(py38general) ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~/nlp$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC3G6NS1UJWnuGAj+IumEF0zLGH8mflQceV5bKdSDJxUgZh9/meanPDUUKfS0pXvn9ReZYf
6Y5h/J1hZsEh2fXVOtoAD1wwFWq6NeMbkhMBAaGQdb6RYf2ZXPlHUNHG/ZJjhFiRILCxJ2AhSxu/jhICgRZsDIirIm/4SSTWKFgNkGme+z67
FszRWpKkrD0931hfJpM4kNLeelfyqFOjBmhGRYxkDMiWhmvZGmbbOJk07OChJHm1QO5oWRR/UYsNrox0YOQvLqjEFyk0df7og8744is5m81+
UI3dEBqPF1CnFPExKVfoyeqArEXvzLtUPb/o47/ajNC8FAhy/pnsU8+/Te0ZbMfJZlWa74cQ7xQ2k= ubuntu@ml-ubuntu20-04-desktop
```

GitHub Repo

- Register a github account, add your public ssh key to the github
- So that you can clone from github repos



Git Operations

- Git is a code version control tools. Every programmer should learn its basic usage.

- git init
- git commit –m message
- git pull
- git checkout
- git pop
- git merge

git add .
git clone
git push
git stash

- .gitignore
- resolve conflicts …

- Learn on your own. Much more to explore here.

Bash

- Running big models generally requires submitting bash scripts which automates the pipeline.
- Basic Bash knowledge
- For Loop
- Parallel Computation
- Passing Arguments
- Redirecting Outputs

Bash

- Running big models generally requires submitting bash scripts which automates the pipeline.

For Loop

```
for VARIABLE in 1 2 3 4 5 .. N
do
     command1
     command2
     commandN
done
```

```
#!/bin/bash
for i in {1..5}
do
   echo "Welcome $i times"
done
```

https://www.cyberciti.biz/faq/bash-for-loop/

Bash

- Running big models generally requires submitting bash scripts which automates the pipeline.

Parallel Computation

```
for stuff in things
do
( something
  with
  stuff ) &
done
wait # for all the something with stuff
```

https://unix.stackexchange.com/questions/103920/parallelize-a-bash-for-loop

Bash

- Running big models generally requires submitting bash scripts which automates the pipeline.

Variables (no space around =)

```
#!/bin/bash
# A simple variable example

myvariable=Hello

anothervar=Fred

echo $myvariable $anothervar
echo

sampledir=/etc

ls $sampledir
```

$1, $2, ...
The first, second, etc command line arguments to the script.

variable=value
To set a value for a variable. Remember, no spaces on either side of =

Quotes " '
Double will do variable substitution, single will not.

variable=$(command)
Save the output of a command into a variable

export var1
Make the variable var1 available to child processes.

```
user@bash: myvar='Hello World'
user@bash: echo $myvar
Hello World
user@bash: newvar="More $myvar"
user@bash: echo $newvar
More Hello World
user@bash: newvar='More $myvar'
user@bash: echo $newvar
More $myvar
user@bash:
```

https://ryanstutorials.net/bash-scripting-tutorial/bash-variables.php

Bash

- Running big models generally requires submitting bash scripts which automates the pipeline.

Passing Arguments

```
echo "Username: $1";
echo "Age: $2";
echo "Full Name: $3";
```

```
sh userReg-positional-parameter.sh john 25 'John Smith'
```

```
while getopts u:a:f: flag
do
    case "${flag}" in
        u) username=${OPTARG};;
        a) age=${OPTARG};;
        f) fullname=${OPTARG};;
    esac
done
echo "Username: $username";
echo "Age: $age";
echo "Full Name: $fullname";
```

```
sh userReg-flags.sh -f 'John Smith' -a 25 -u john
```

https://linuxize.com/post/bash-functions/

Bash

- Running big models generally requires submitting bash scripts which automates the pipeline.

Redirecting Outputs

`n>`

```
command > file
```

```
command 1> file
```

The same

```
command 2> error.txt 1> output.txt
```

```
command > file 2>&1
```

Have error messages sent to the same file as standard output

command >> output.txt Append instead of overwrite

https://linuxize.com/post/bash-functions/

Homework

- https://github.com/thunlp/BMCourse/tree/main/exercises/L1_Word2Vec_hpsearch

- Install virtual environment
- git clone this repo https://github.com/OlgaChernytska/word2vec-pytorch
- Training a simple word2vec
- Write some simple bash script to automatically search for hyperparamters. Automatically summarize the output {hyperparameters, validation loss} to one result file.

Academic Websites

- Google Scholar: http://scholar.google.com/
- ACM Portal: http://dl.acm.org/
- Association of Computational Lingusitics
  - ACL Anthology: http://www.aclweb.org/anthology/
  - ACL、EMNLP、NAACL、COLING、TACL
- Other conferences
  - ICLR、NeurIPS、ICML、AAAI、CVPR、…

Q&A session

- Each Tuesday and Thursday 10:00-11:00 is Q&A time
  - TAs will be in the same Tencent Meeting as today to answer questions about exercises and projects.

Thanks for listening

THUNLP