



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



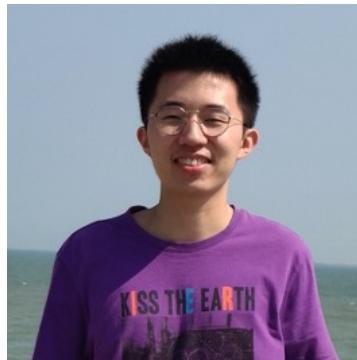
- TA: [Weize Chen](#)
- PhD student
- Research Interest: Architecture of Big Models





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

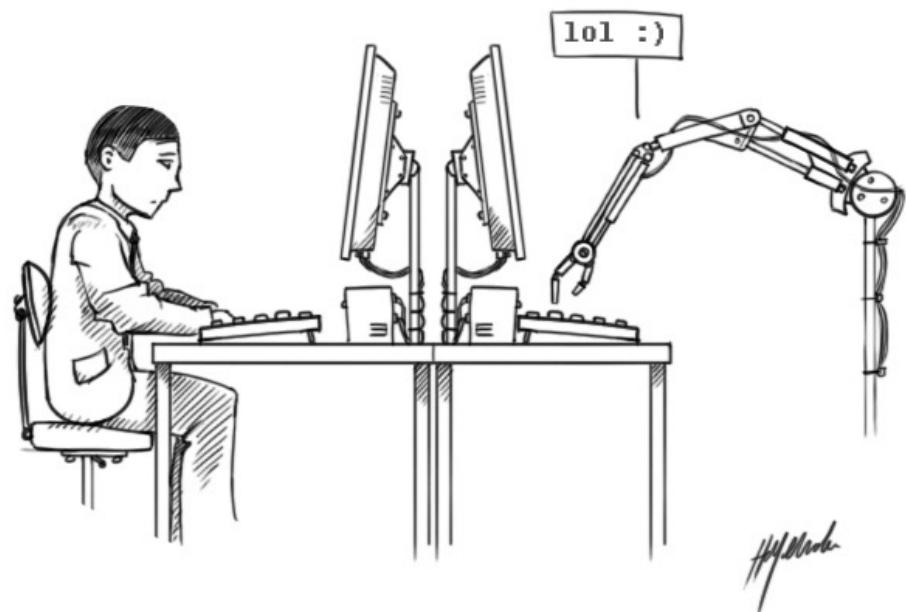
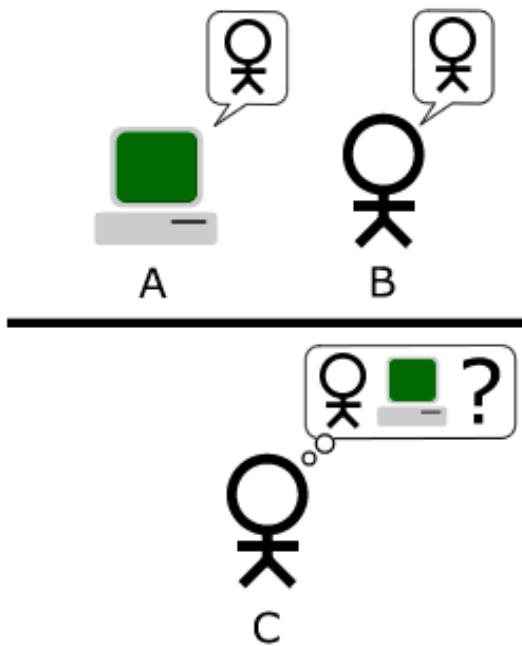
Why is NLP Important?

THUNLP



Scientific Impact of NLP

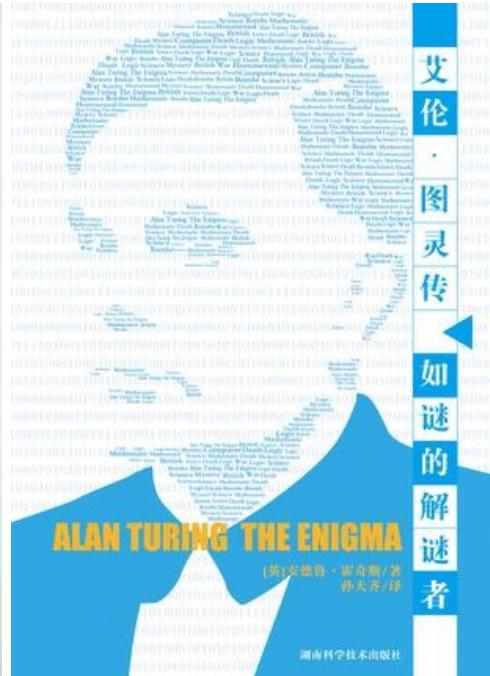
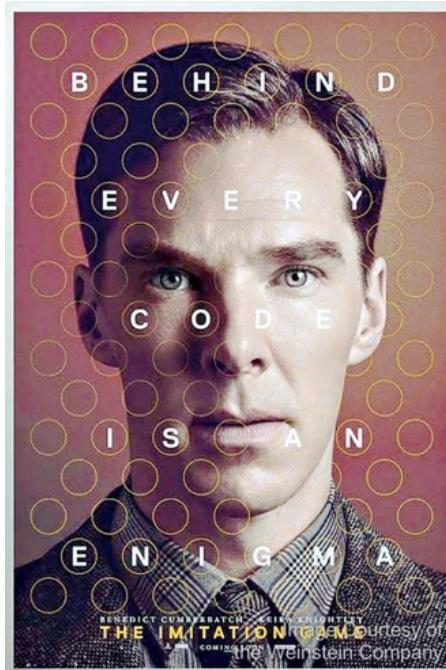
- Turing Test: A test of machine ability to exhibit intelligent behavior indistinguishable from a human
- Language is the communication tool in the test





Scientific Impact of NLP

- Origin Version: Imitation Game



Turing, Alan, 1912-2004. *艾伦·图灵*, 1906-1954.

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

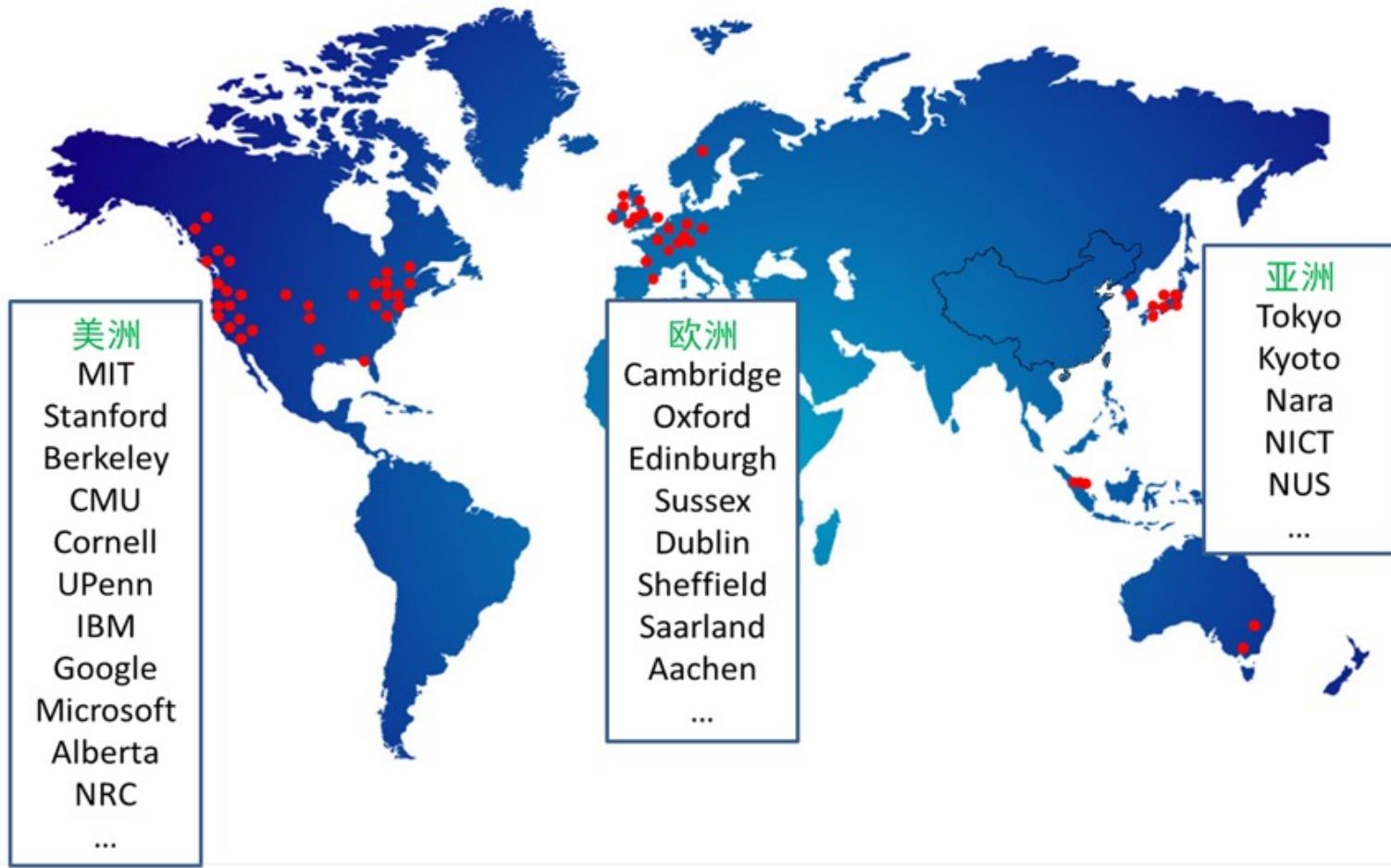


Q: Who was presidentially pardoned on September 8, 1974?
A: Nixon.



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

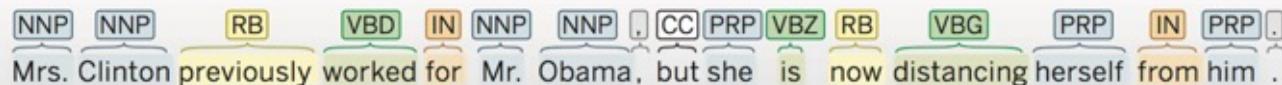
THUNLP



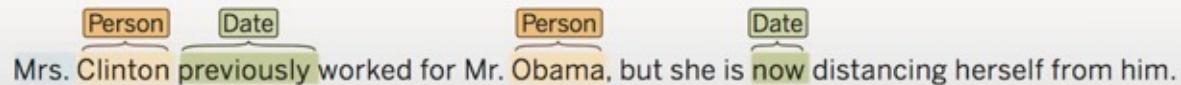
NLP Tasks

- Basic Tasks of NLP

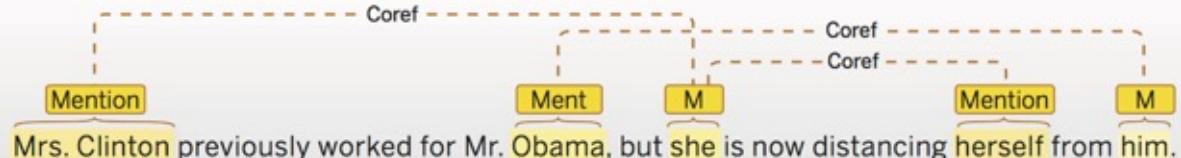
Part of speech:



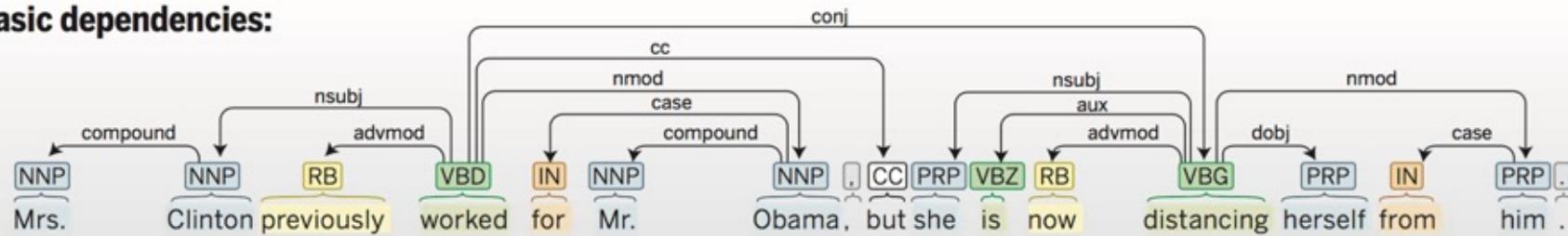
Named entity recognition:



Co-reference:



Basic dependencies:





Search Engines and Ads



Structural Knowledge



write



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

head entity

relation

tail entity



Knowledge Graph

The background of the slide is a dark, semi-transparent network graph. It consists of numerous small, white or light-gray dots representing nodes, connected by thin, dark lines representing edges. This visual metaphor represents the complex, interconnected nature of a knowledge graph.

The Knowledge Graph



KG Application: Question Answering

- KG provides the knowledge facts of a question

The screenshot shows the WolframAlpha search interface. The query "how big is China" is entered in the search bar. Below the search bar, there are several small icons representing different types of data (e.g., map, video, chart). To the right of the search bar are links for "Examples" and "Random". A note below the search bar states: "Assuming 'how big' is international data | Use as referring to socioeconomic data or referring to species or referring to administrative divisions instead". Another note says: "Assuming total area | Use population instead".

Input interpretation:

China total area

Result:

$9.597 \times 10^6 \text{ km}^2$ (square kilometers) (world rank: 4th)

Show non-metric

Unit conversions:

$9.597 \times 10^{12} \text{ m}^2$ (square meters)

3.705 million mi² (square miles)

$1.033 \times 10^{14} \text{ ft}^2$ (square feet)

Comparisons as area:

$\approx 0.96 \times$ total area of Canada ($9.98467 \times 10^6 \text{ km}^2$)

$\approx 0.996 \times$ total area of the United States ($9.63142 \times 10^6 \text{ km}^2$)

\approx largest extent of the Roman Empire ($\approx 9 \text{ Mm}^2$)



Machine Reading

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

The Right Honourable
John Curtin

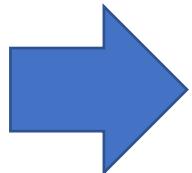


14th Prime Minister of Australia
Elections: 1937, 1940, 1943

Personal details

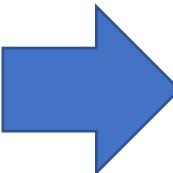
Born John Joseph Curtin
8 January 1885
Creswick, Colony of Victoria
British Empire

Died 5 July 1945 (aged 60)
Canberra, Australia



per: date_of_birth per: city_of_birth

John Curtin (1885-1945), prime minister and journalist, was born on 8 January 1885 at Creswick, Victoria, eldest of four children of Irish-born parents John Curtin and his wife Catherine (Kate) Agnes, née Bourke. ...



Learn how to extract slots from free text:

was born on <DATE>
per: date_of_birth

born ... at <LOCATION>
per: city_of_birth

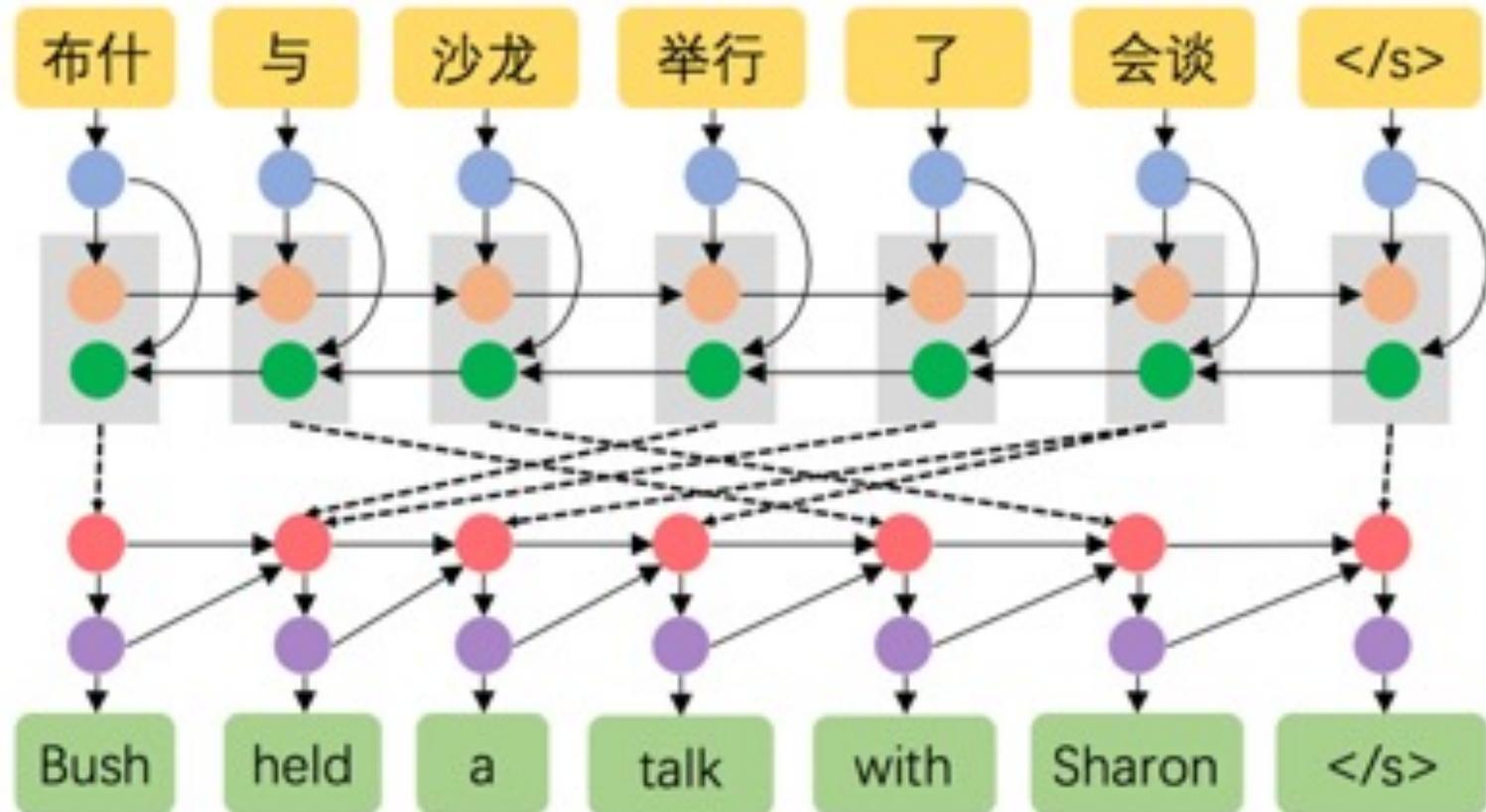


Personal Assistant





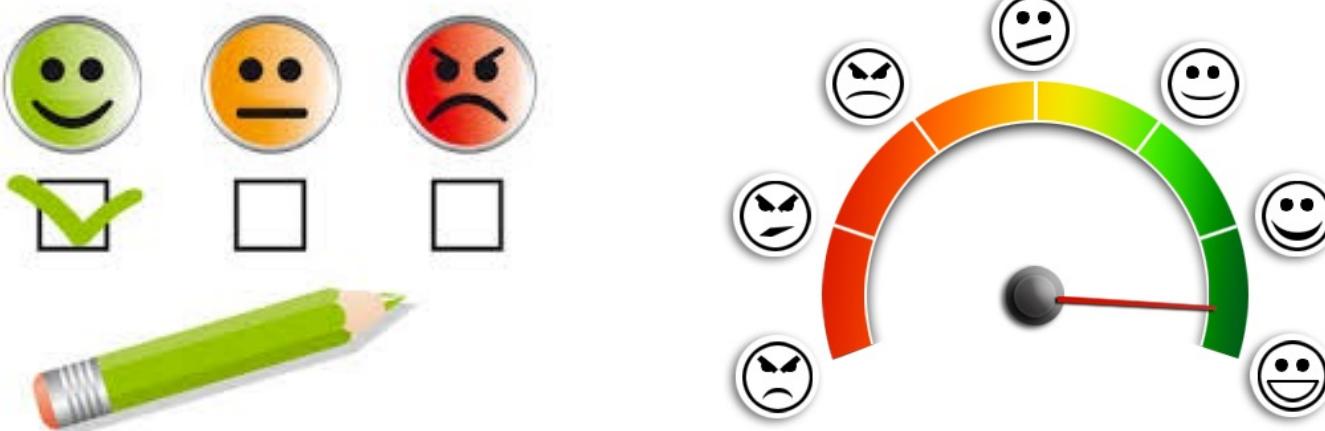
Machine Translation





Sentiment Analysis and Opinion Mining

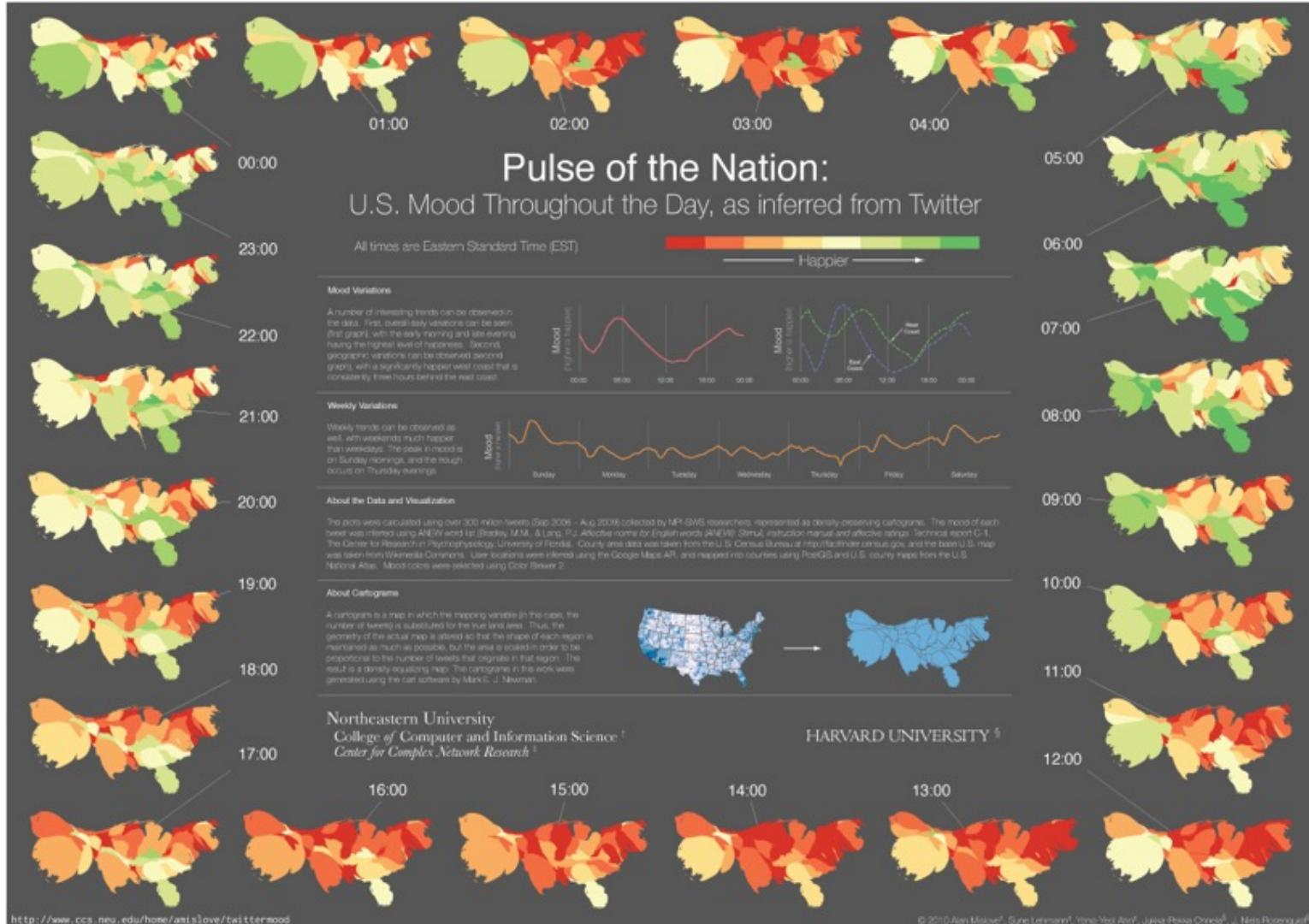
- Infer personal states via text or speech
 - Including opinions, emotions, ...



- Detect opinion holders and targets



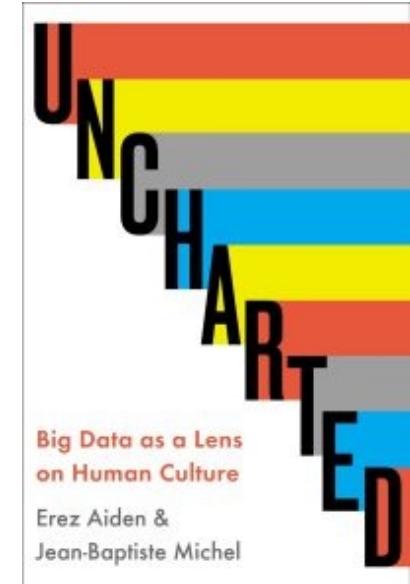
Sentiment Analysis and Opinion Mining





Computational Social Science

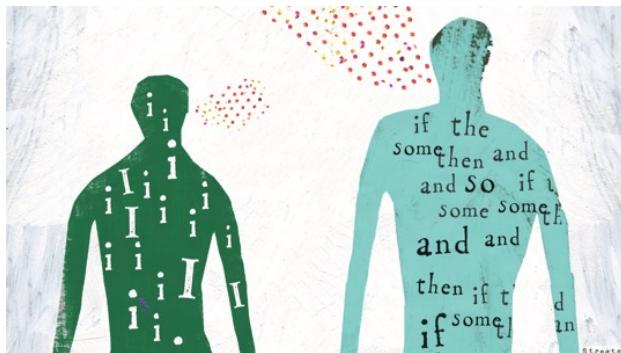
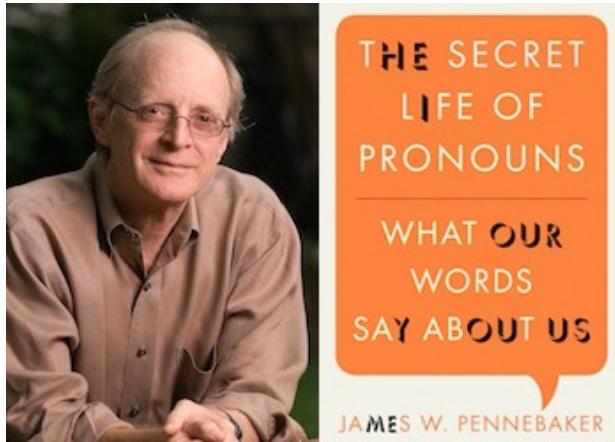
- Culturomics: www.culturomics.org
- Harvard researchers use keywords over Google Books (5 million books from 1800 to 2000) to study the evolution of human culture
- Google Book N-gram: books.google.com/ngrams





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 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

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 no 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.

LIWC results from input text

LIWC results from personal text and formal writing for comparison

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**
 - $\text{WR}(\text{Star}) \simeq \text{WR}(\text{Sun})$
 - $\text{WR}(\text{Motel}) \simeq \text{WR}(\text{Hotel})$
- Infer word **relation**
 - $\text{WR}(\text{China}) - \text{WR}(\text{Beijing}) \simeq \text{WR}(\text{Japan}) - \text{WR}(\text{Tokyo})$
 - $\text{WR}(\text{Man}) \simeq \text{WR}(\text{King}) - \text{WR}(\text{Queen}) + \text{WR}(\text{Woman})$
 - $\text{WR}(\text{Swimming}) \simeq \text{WR}(\text{Walking}) - \text{WR}(\text{Walk}) + \text{WR}(\text{Swim})$



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

- $\text{similarity}(\text{star}, \text{sun}) = (\nu_{\text{star}}, \nu_{\text{sun}}) = 0$
- 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

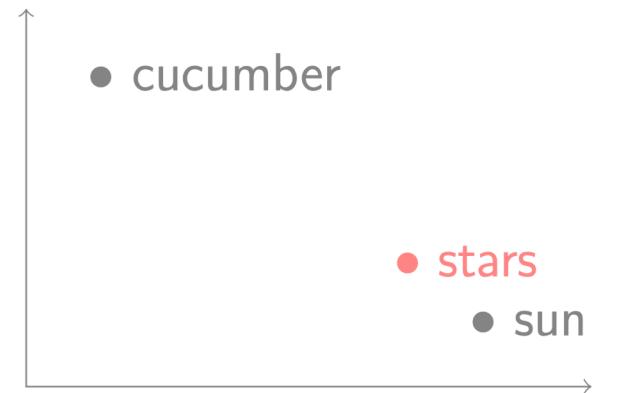


Co-Occurrence Counts

- 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



stars	shining	bright	trees	dark	look
	38	45	2	27	12



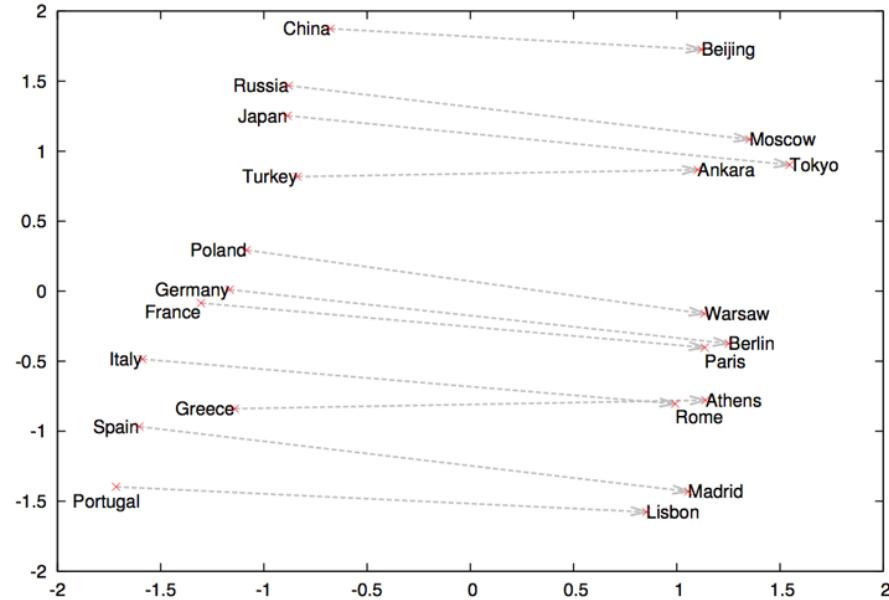
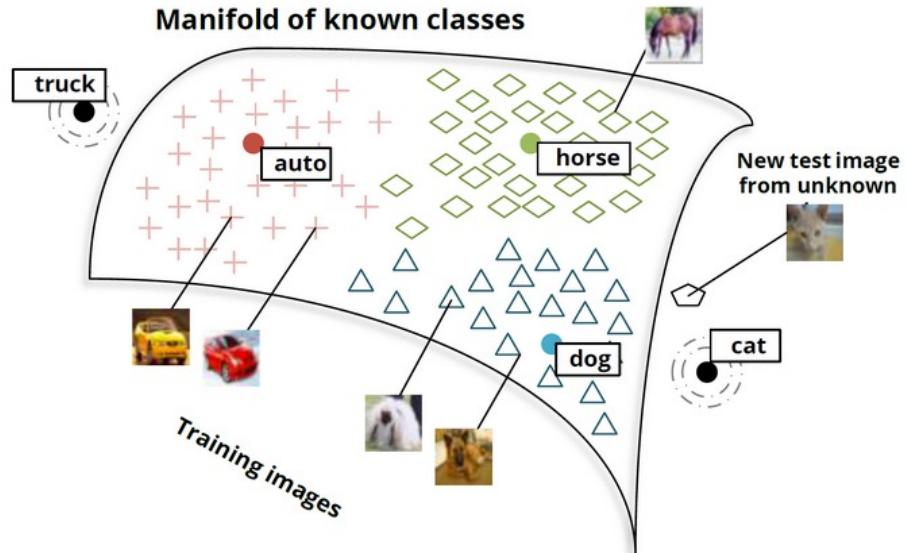
Problems of Count-Based Representation

- 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

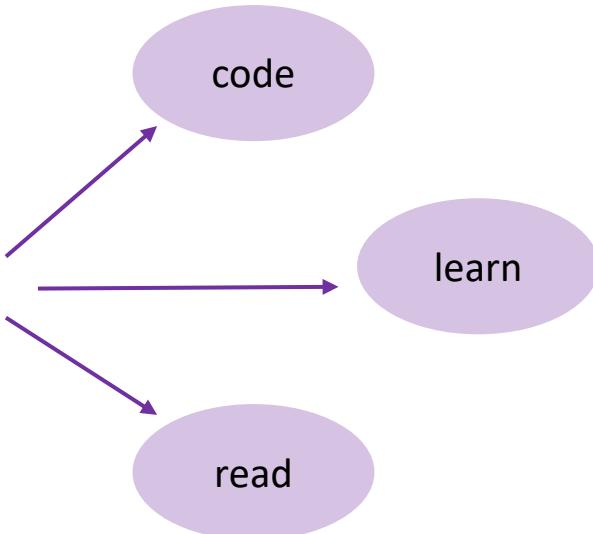


Language Model

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

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

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:

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

- Compute conditional probability of an upcoming word w_n :

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

- 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

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



N-gram Model

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

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

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



N-gram Model

- Markov assumption

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

- And we have:

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



Andrei Markov

- Simplifying Language Model

- Bigram (N-Gram, N = 2)

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

- Trigram (N-Gram, N = 3)

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



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)

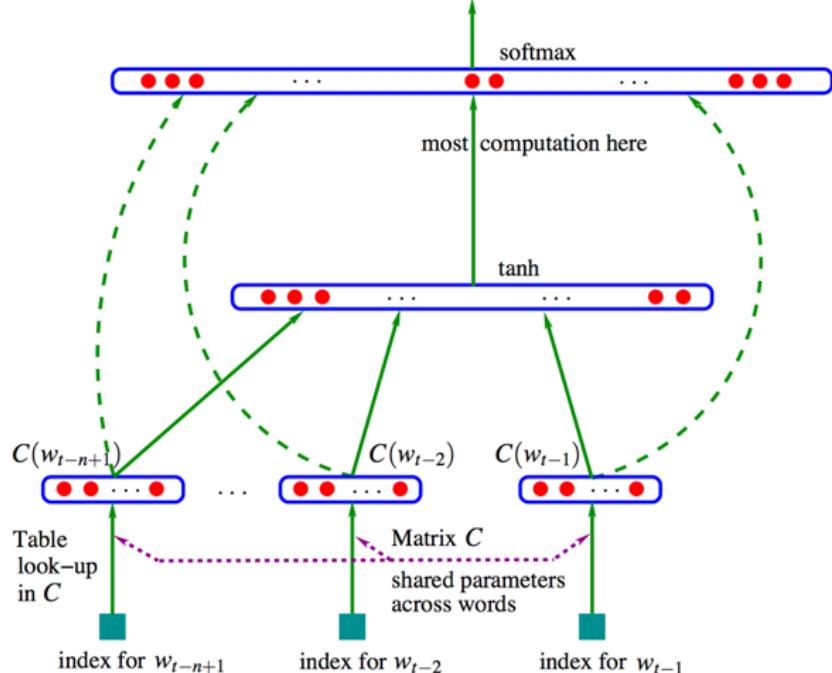


Neural Language Model

Journal of Machine Learning Research 3 (2003) 1137–1155

Submitted 4/02; Published 2/03

$$i\text{-th output} = P(w_t = i \mid \text{context})$$



Cited by 8858



A Neural Probabilistic Language Model

Yoshua Bengio

Réjean Ducharme

Pascal Vincent

Christian Jauvin

Département d'Informatique et Recherche Opérationnelle

Centre de Recherche Mathématiques

Université de Montréal, Montréal, Québec, Canada

BENGIOY@IRO.UMONTREAL.CA

DUCHARME@IRO.UMONTREAL.CA

VINCENTP@IRO.UMONTREAL.CA

JAUINC@IRO.UMONTREAL.CA

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 if a sequence of words that have never been seen before gets high probability if it is made of words that are similar (in the sense of having many neighbors) 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 $100,000^{10} - 1 = 10^{50} - 1$ 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-

©2003 Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin.



Big Model Basics

The Trip to Big Models

Xu Han

THUNLP



The Trip to Big PLMs

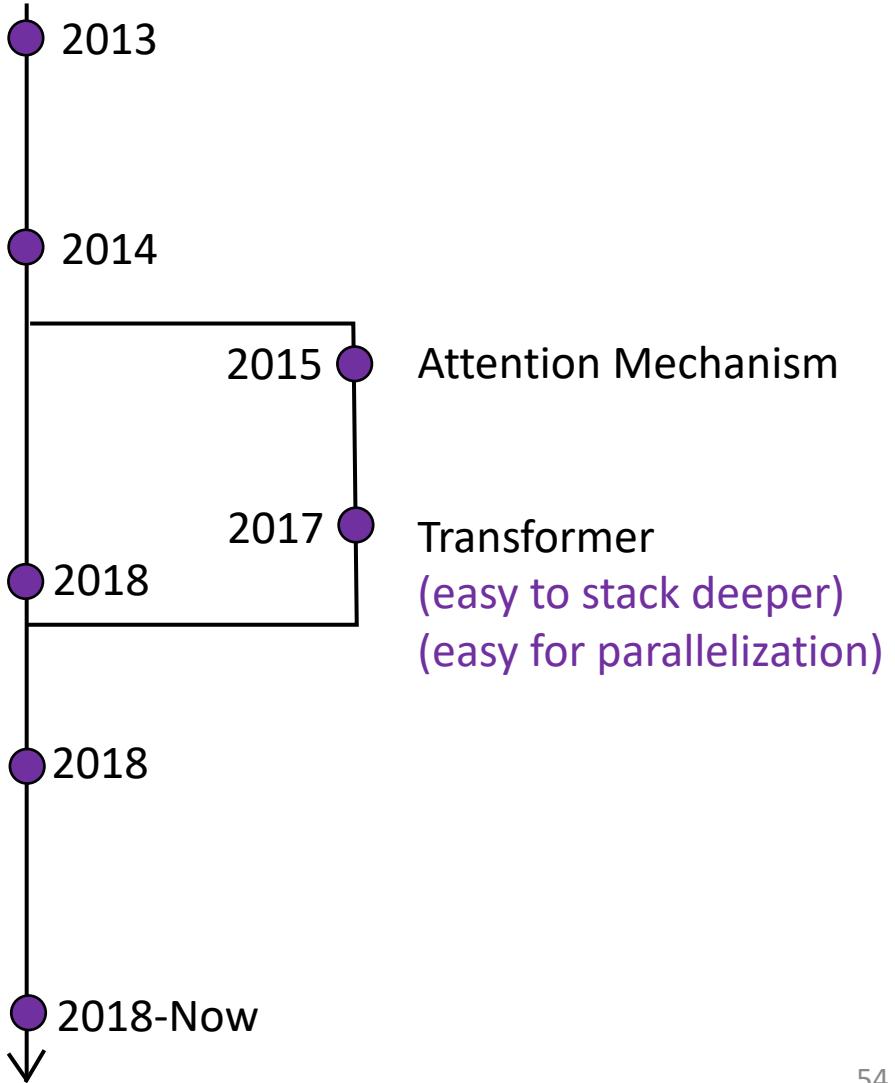
Word-level Modeling (Word2Vec)
(self-supervised learning)

Sequence-level Modeling (RNN)
(supervised learning)

Contextualized Embedding (ELMo)
(self-supervised learning)

Pre-trained Language Model (BERT ...)
(self-supervised learning)

Language Model Scaling (GPT3 ...)
(self-supervised learning)





Big Model Basics

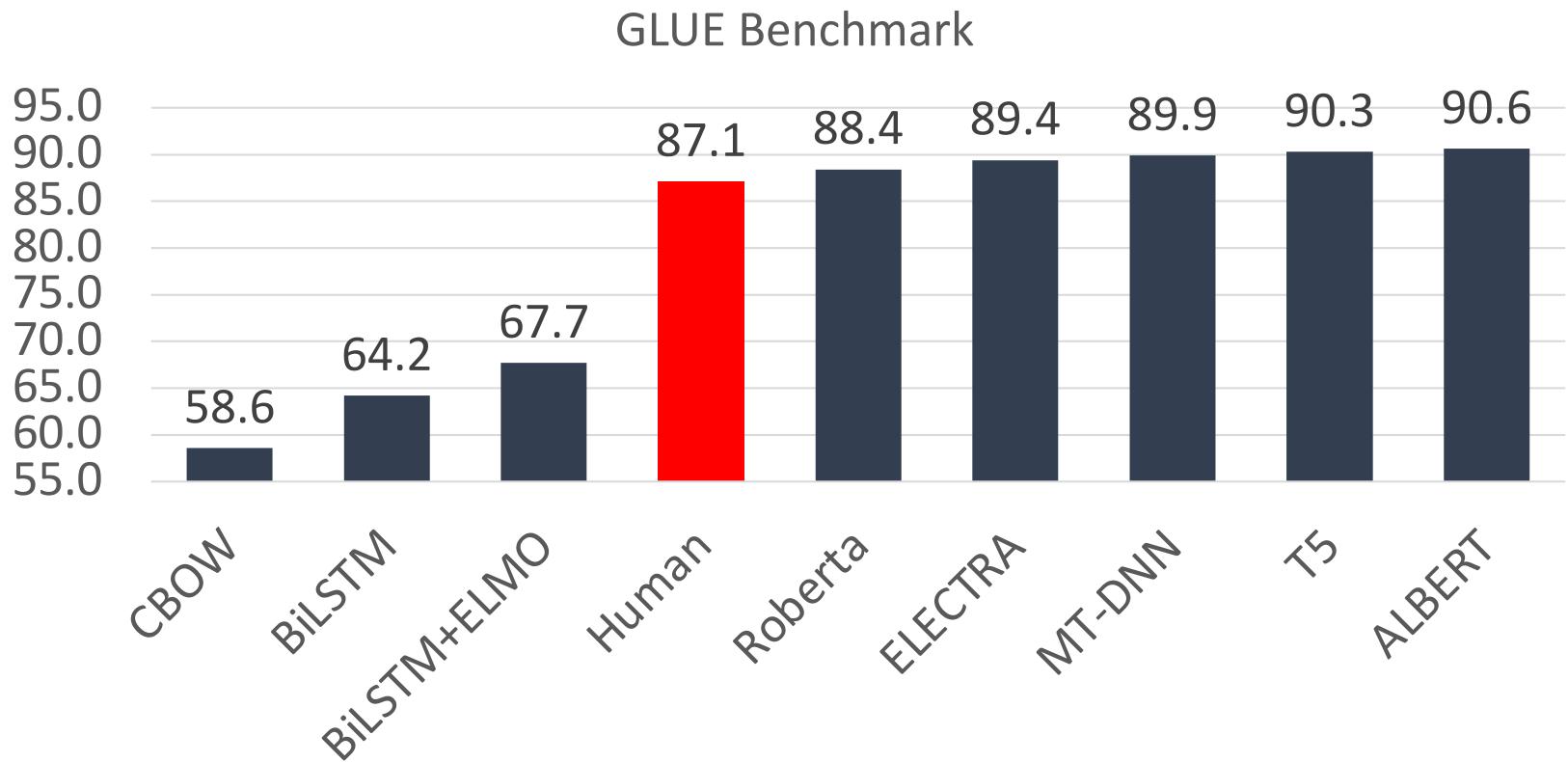
Why are Big Models Important

THUNLP



Why PLMs

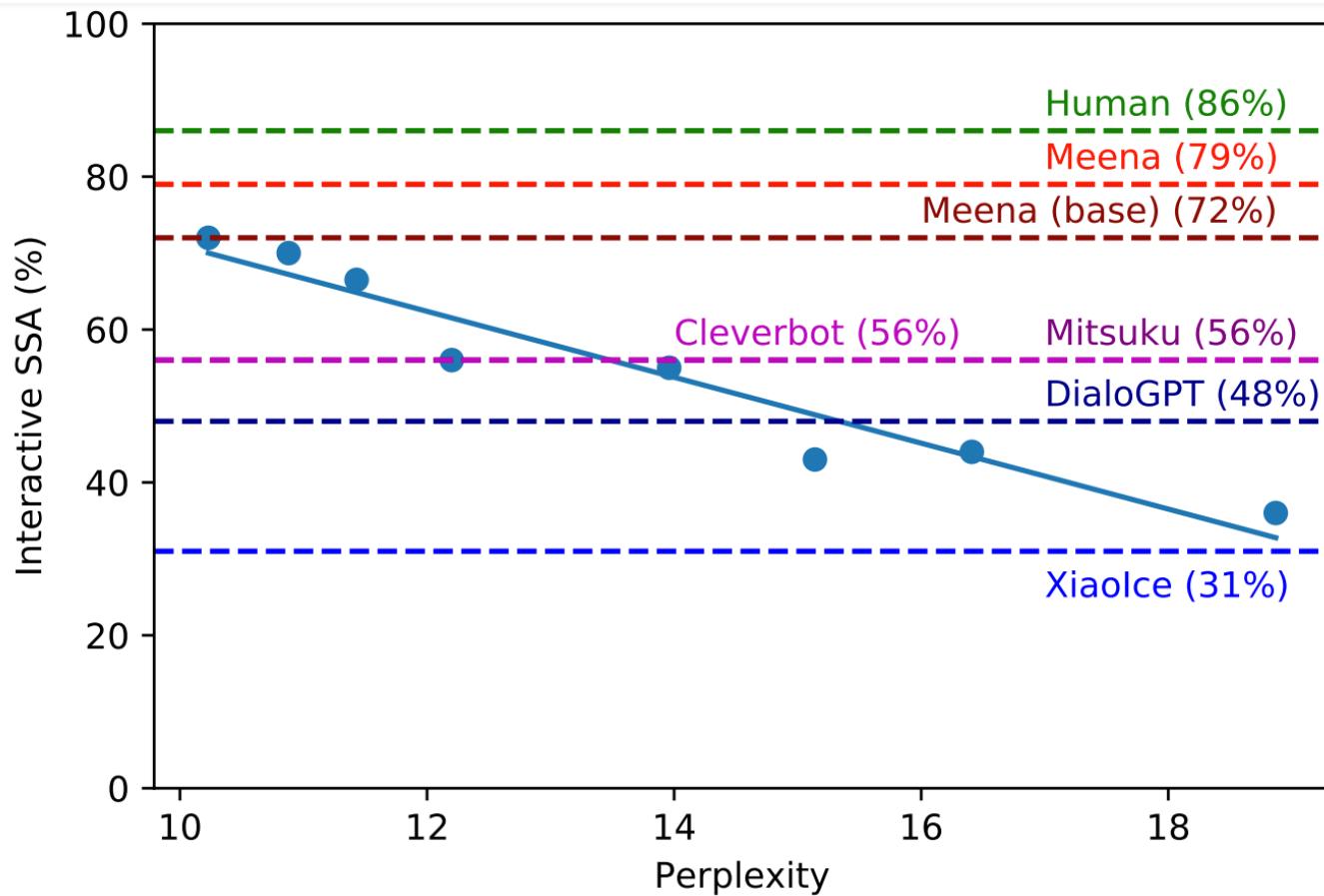
The results of the pre-trained language model on GLUE surpass human level, reflecting the ability of **language understanding**





Why PLMs

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



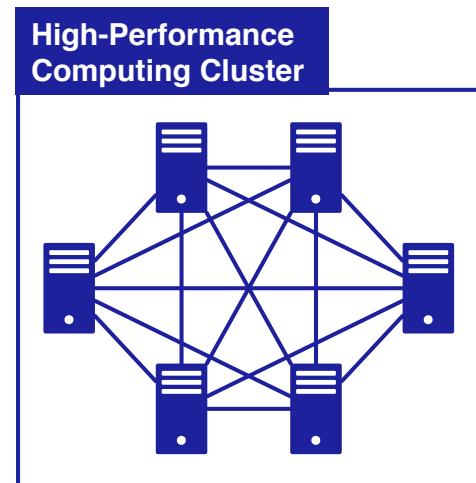
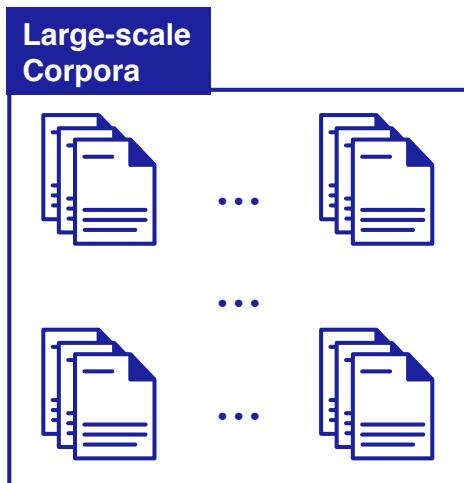
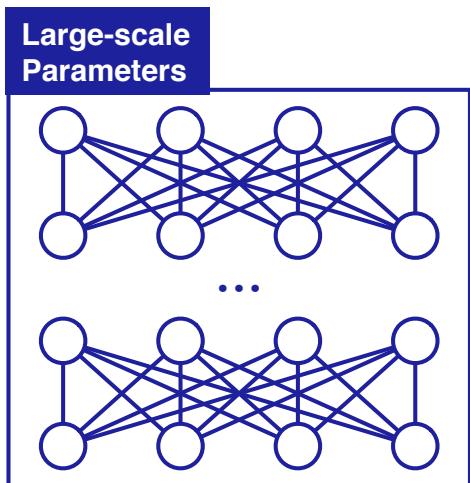


Why Big Models

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

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

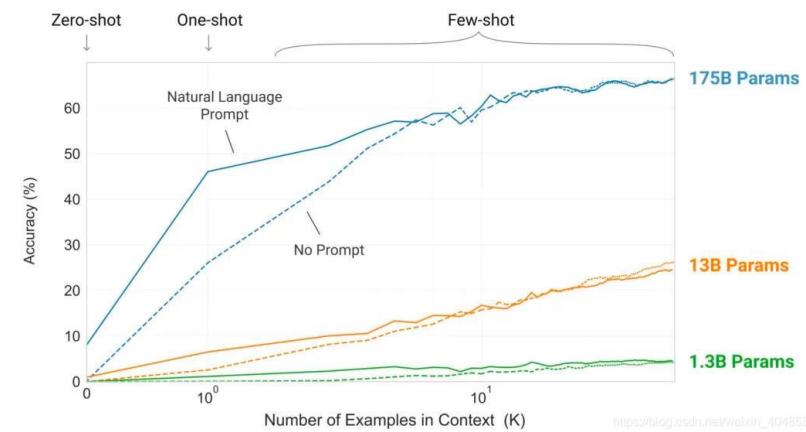
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.





Why Big Models

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

The image displays two side-by-side screenshots illustrating GPT-3's capabilities.

Left Side (Stockfish 11+ WASMX):

- Chessboard:** A standard 8x8 chessboard with pieces in their starting positions. A grey arrow points from the white pawn at e4 to the black pawn at e5, indicating a move.
- Move List:** A list of 15 moves from a game analysis:
 1. e4 e5
 2. $\mathbb{Q}f3$ $\mathbb{Q}c6$
 3. $\mathbb{Q}c4$ $\mathbb{Q}f6$
 4. O-O
 5. $\mathbb{Q}c5$
 6. $\mathbb{Q}c3$ d6
 7. O-O
 8. f4
 9. d6
 10. $\mathbb{Q}f3$
 11. $\mathbb{Q}e7$
 12. h6
 13. $\mathbb{W}e1$
 14. d5
 15. exd5

Right Side (GPT-3 Sandbox):

- Text Input:** A text input field containing "Build Keras Models".
- Generate Model:** A blue button labeled "Generate Model".

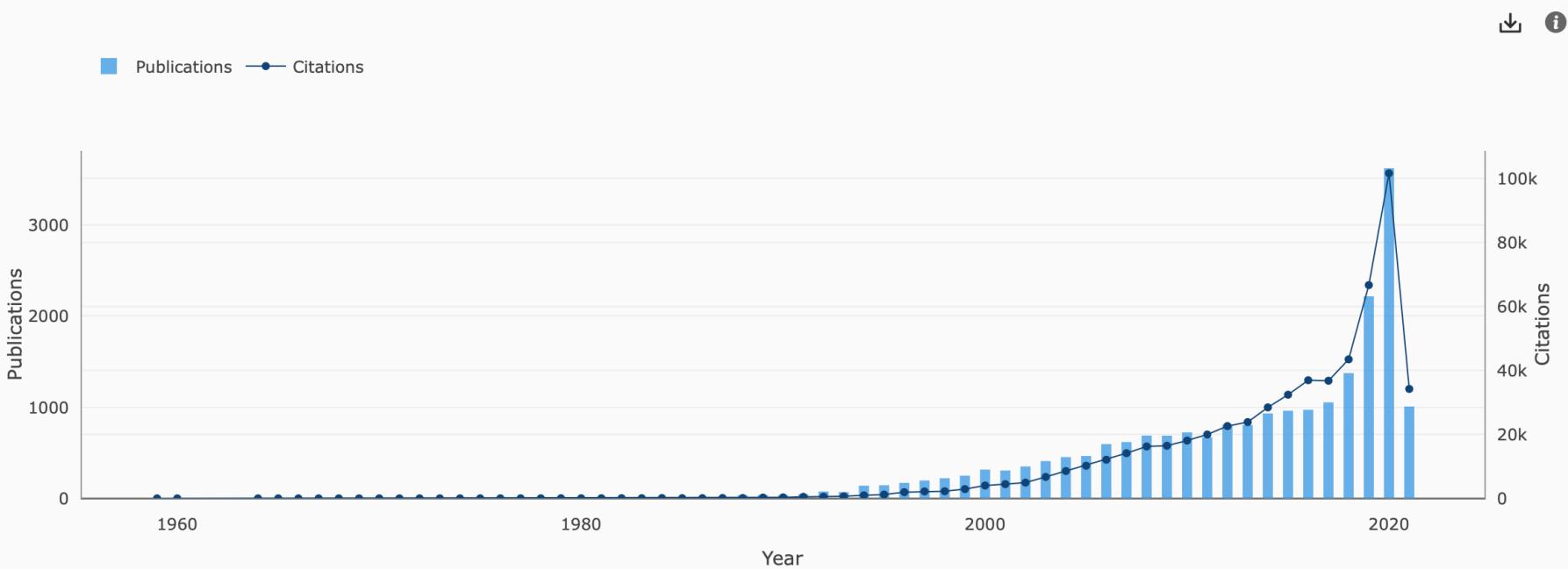


Why Big Models

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

Language model related research has also grown rapidly after 2018

Publications & Citations Over Time





Big Model Basics

Paradigms Behind Big Models

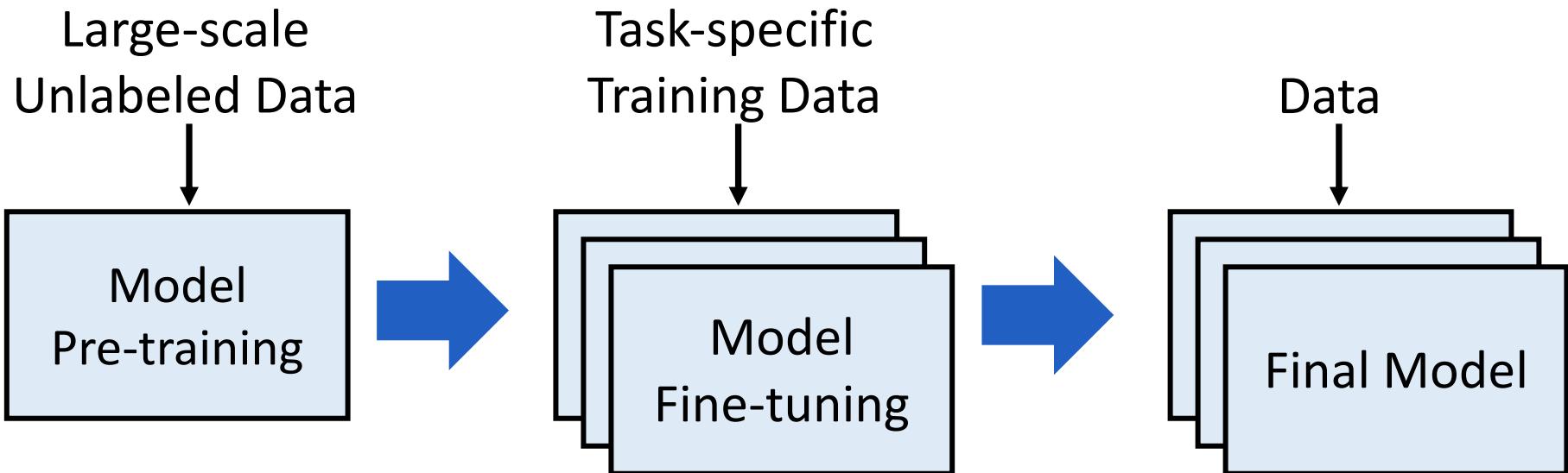
THUNLP



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**

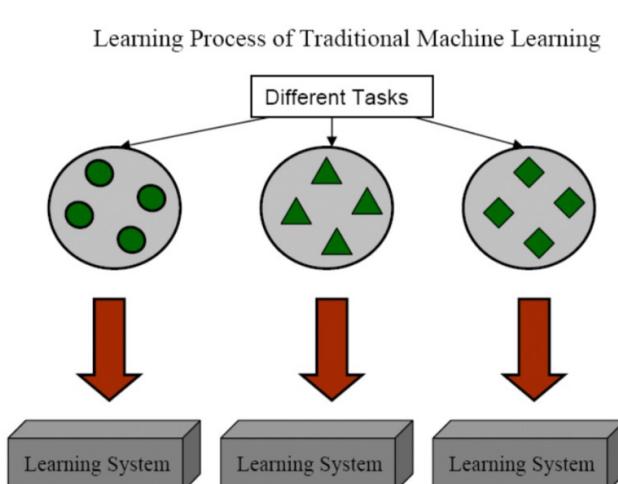




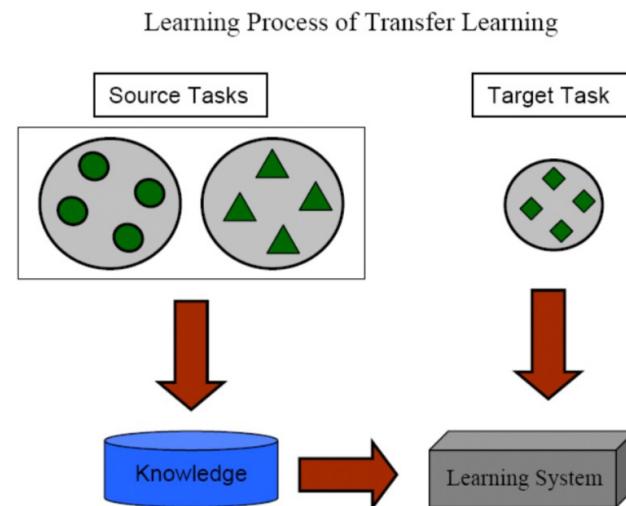
Paradigms behind Big Models

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



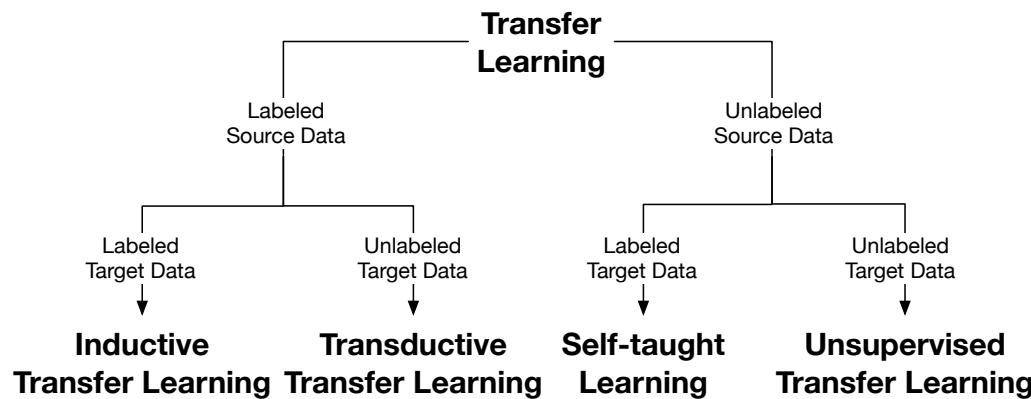
(b) Transfer Learning



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
<i>Instance-transfer</i>	✓	✓	
<i>Feature-representation-transfer</i>	✓	✓	✓
<i>Parameter-transfer</i>	✓		
<i>Relational-knowledge-transfer</i>	✓		

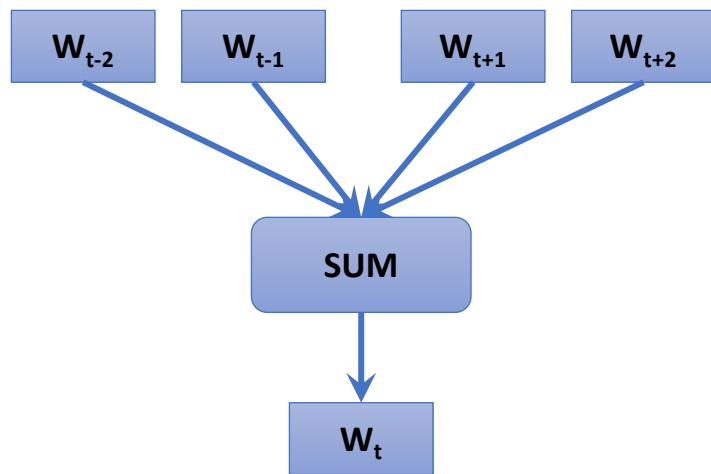


Paradigms behind Big Models

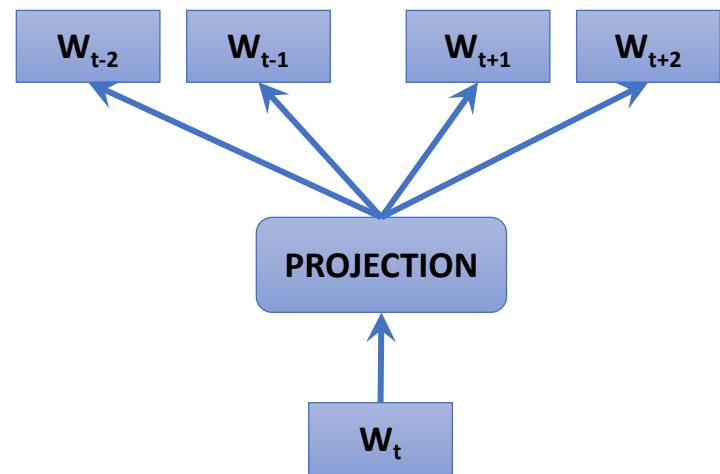
Word embeddings can be seen as simple pre-trained language models, such as CBOW and Skip-Gram

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



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

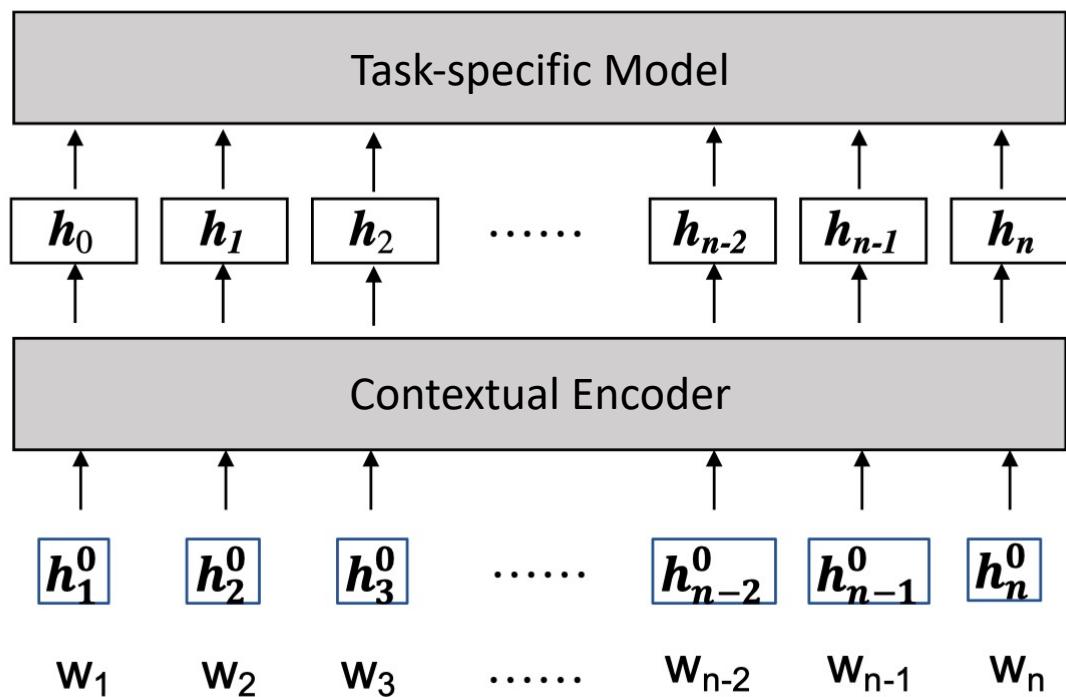
Hard to detect antonym

- 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



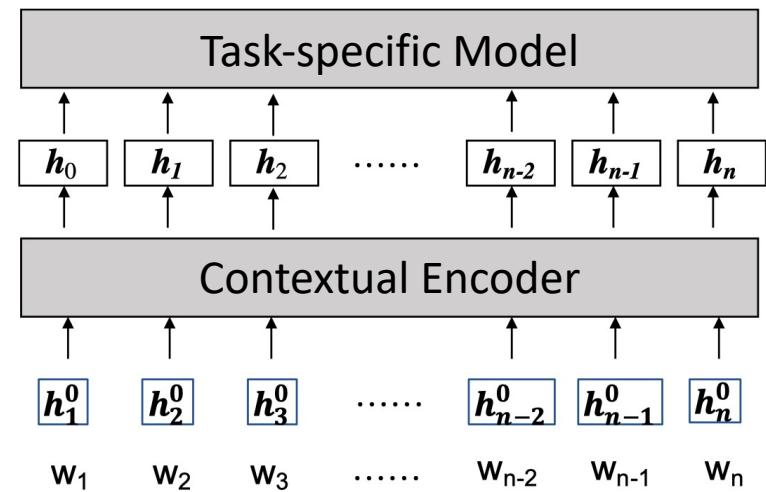
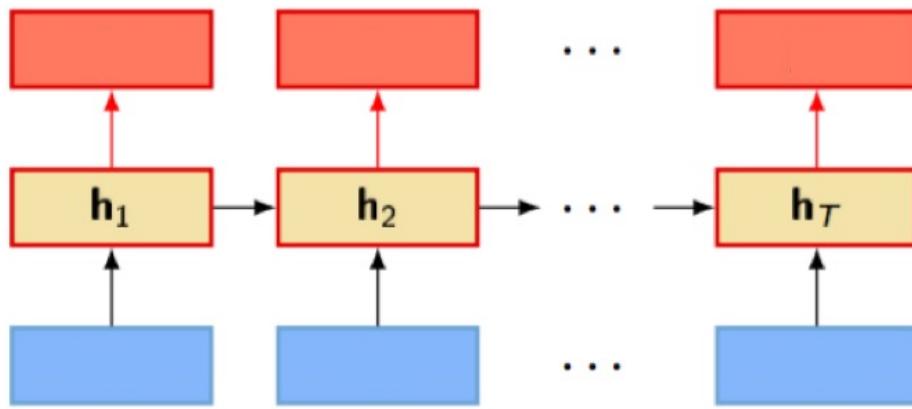


Paradigms behind Big Models

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

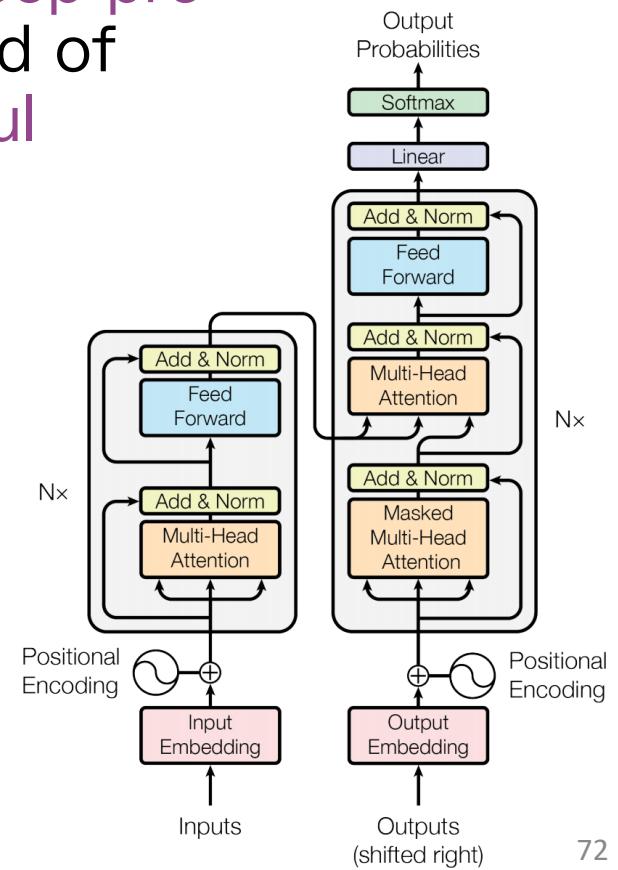
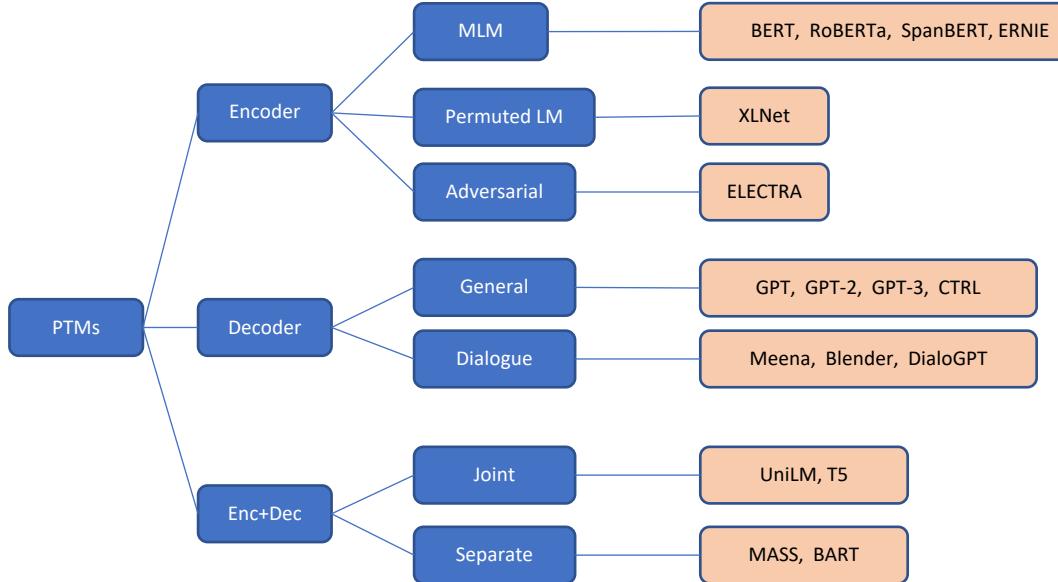
ULMFiT: Universal Language Model Fine-tuning for Text Classification



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**





Big Model Basics

Typical Case of Big Models

THUNLP



Typical Case of Big Models

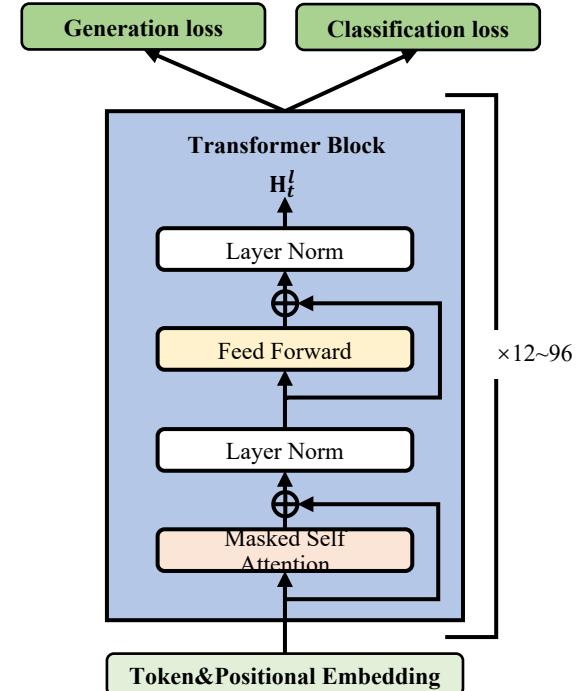
GPT:

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)

store
↑
the man went to the [MASK] to buy a [MASK] of milk
gallon
↑

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

Mark Twain wrote **The Million Pound Bank Note** in 1893.

Input for Common NLP tasks:



Input for Entity Typing:



Input for Relation Classification:





Big Model Basics Demos

THUNLP



Demos of Big Models

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

The following is a conversation with an AI assistant. The assistant is helpful, creative, clever, and very friendly.

Human: Hello, who are you?
AI: I am an AI created by OpenAI. How can I help you today?
Human:

Chat X Save View code Share ...

Mode grid list card

Model text-davinci-002

Temperature 0.9

Maximum length 150

Stop sequences
Enter sequence and press Tab

Human: x

AI: x

Top P 1

Frequency penalty 0

Presence penalty 0.6

Submit Cancel Copy Downvote Upvote

56 Best of 1



Demos of Big Models

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

bles Playground

⚡ Upgrade

?

Help

S

Tsinghua

Playground

Q&A X Save View code Share ...

A: Dwight D. Eisenhower was president of the United States in 1955.

Q: Which party did he belong to?
A: He belonged to the Republican Party.

Q: What is the square root of banana?
A: Unknown

Q: How does a telescope work?
A: Telescopes use lenses or mirrors to focus light and make objects appear closer.

Q: Where were the 1992 Olympics held?
A: The 1992 Olympics were held in Barcelona, Spain.

Q: How many squigs are in a bonk?
A: Unknown

Q:

Mode ☰ ⬇️ ☰

Model text-davinci-002

Temperature 0

Maximum length 100

Stop sequences Enter sequence and press Tab

Top P 1

Frequency penalty 0

Submit ↻ ⟳

223

79



Demos of Big Models

- Scientific plotting with CodeX (175B)

The screenshot shows the CodeX Verification - Colaboratory playground interface. At the top, there are navigation links: Overview, Documentation, Examples, and Playground (which is highlighted). On the right, there are buttons for Upgrade, Help, and a user profile for Tsinghua. The main area is titled "Playground" and contains a text input field with the placeholder "Natural language to python". Below the input field, a code completion suggestion is shown:

```
1 /* Create a JavaScript dictionary of 5 countries and capitals: */
```

To the right of the input field are several configuration options:

- Mode:** A dropdown menu with three icons.
- Model:** A dropdown menu set to "code-davinci-002".
- Temperature:** A slider set to 0.
- Maximum length:** A slider set to 715.
- Stop sequences:** An input field with placeholder "Enter sequence and press Tab".
- Top P:** A slider set to 1.
- Frequency penalty:** A slider set to 0.
- Presence penalty:** A slider set to 0.
- Best of:** A slider set to 1.

At the bottom of the playground interface, there is a message: "The Codex models are currently in private beta. Usage is free during this period. [Learn more.](#)"

At the very bottom of the page, there are buttons for "Submit", "Cancel", and "Reset". To the right, there is a status bar showing "1 Python" and an "Inject start text" button.



Demos of Big Models

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





Demos of Big Models

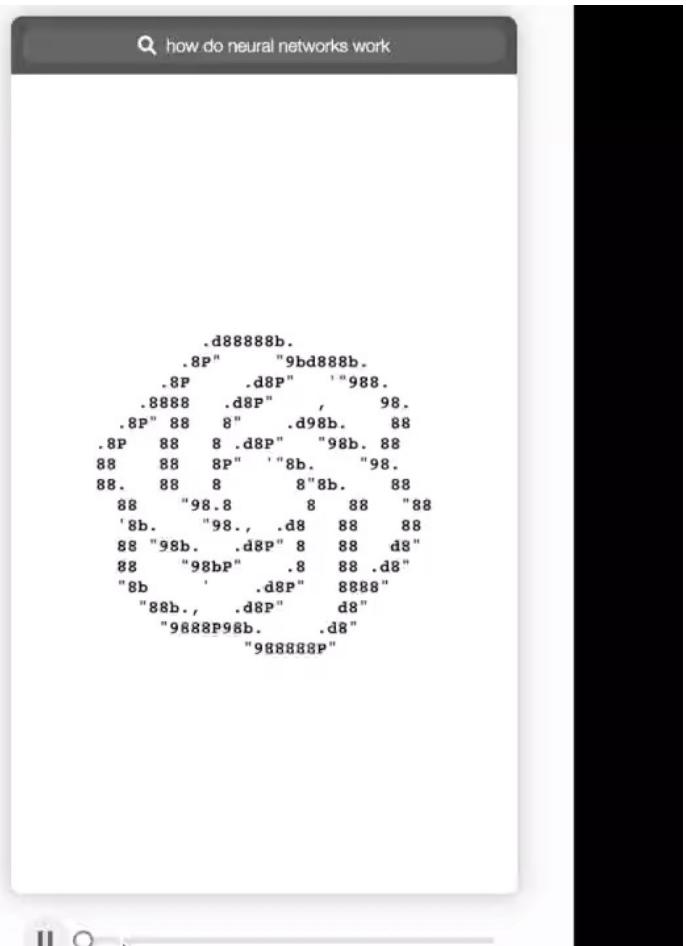
- Web-based question answering with WebGPT



How do neural networks work?

In its most basic form, a neural network has two layers: an input layer and an output layer^[3]. The output layer is the component of the neural network that makes predictions^[3]. In a feedforward network, information flows through the network in the following way: patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units^[1]. The network learns by a feedback process called backpropagation, which involves comparing the output a network produces with the output it was meant to produce, and using the difference between them to modify the weights of the connections between the units in the network, working from the output units through the hidden units to the input units, going backward^{[2][4]}. Over time, backpropagation causes the network to learn, reducing the difference between actual and intended output to the point where the two exactly coincide, so the network figures things out exactly as it should^[2].

1. How neural networks work - A simple introduction (www.explainthatstuff.com) ↗
2. How neural networks work - A simple introduction (www.explainthatstuff.com) ↗
3. How Do Neural Networks Really Work? | Nick McCullum (nickmccullum.com) ↗
4. How Do Neural Networks Really Work? | Nick McCullum (nickmccullum.com) ↗





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

The screenshot shows a cloud host management interface with two main panels.

Left Panel (Login Information):

- Host ID: MC-1x3090-01
- Status: 实时
转包月
- 用户名: ubuntu
- 密码: [REDACTED]
- IP地址: 36.103.203.72
- 操作按钮: 复制 (Copy)
- 友情提示: 主机创建成功5分钟后, 可通过第三方可视化工具进行远程连接。
- 关闭 (Close) 按钮

Right Panel (Server Details):

- Host ID: -区
- 操作系统: Ubuntu20.04-桌面版...
- 资源类型: RTX3090
- GPU信息: 1卡 24GB显存
- 配置规格: 12核 32GB内存
- 系统存储: 标准云盘 80GB
- 带宽: 按流量 25Mbps
- 开发环境: 操作系统: Ubuntu20.04-桌面版...
- 操作菜单 (右侧):
 - 登录信息 (highlighted with a red arrow)
 - WEB SSH
 - 文件传输
 - 重启
 - 关机
 - 调整带宽
 - 调整配置
 - 重装系统



ssh login

Follow the instruction on web learning to login to the server

```
00:48:55 ➔ x ➔ 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)
```

```
System information as of Sun Jun 26 13:40:24 UTC 2022
System load: 0.0 Processes: 352
Usage of /: 49.9% of 77.36GB Users logged in: 0
Memory usage: 2% IPv4 address for enp3s0: 10.0.1.226
Swap usage: 0%
* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
=====
+ 尊敬的用户您好:
+   服务器中已经部署好了TensorFlow、Pytorch以及Opencv环境:
+     加载Tensorflow2.4.1环境, 执行: conda activate Tensorflow-gpu
+     加载Pytorch1.8.1环境, 执行: conda activate Pytorch-gpu
+     加载Opencv环境, 执行: conda activate Opencv
+   图形界面使用:
+     已预装TurboVNC供用户访问图形界面,TurboVNC与其它图形软件冲突, 安装
+     其它图形软件会导致TurboVNC黑屏问题。
+       启动VNC服务,执行: vncserver :1
+       结束VNC服务,执行: vncserver -kill :1
+       访问图形界面需要下载TurboVNC客户端, 下载地址:
+         https://sourceforge.net/projects/turbovnc/files/
+       登录图形界面: 打开TurboVNC客户端输入 IP:1 后登录。
=====

Last login: Sun Jun 26 10:12:53 2022 from 166.111.138.87
ubuntu@ml-ubuntu20-04-desktop-v2-6-32gb-25m:~$ █
```



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
```

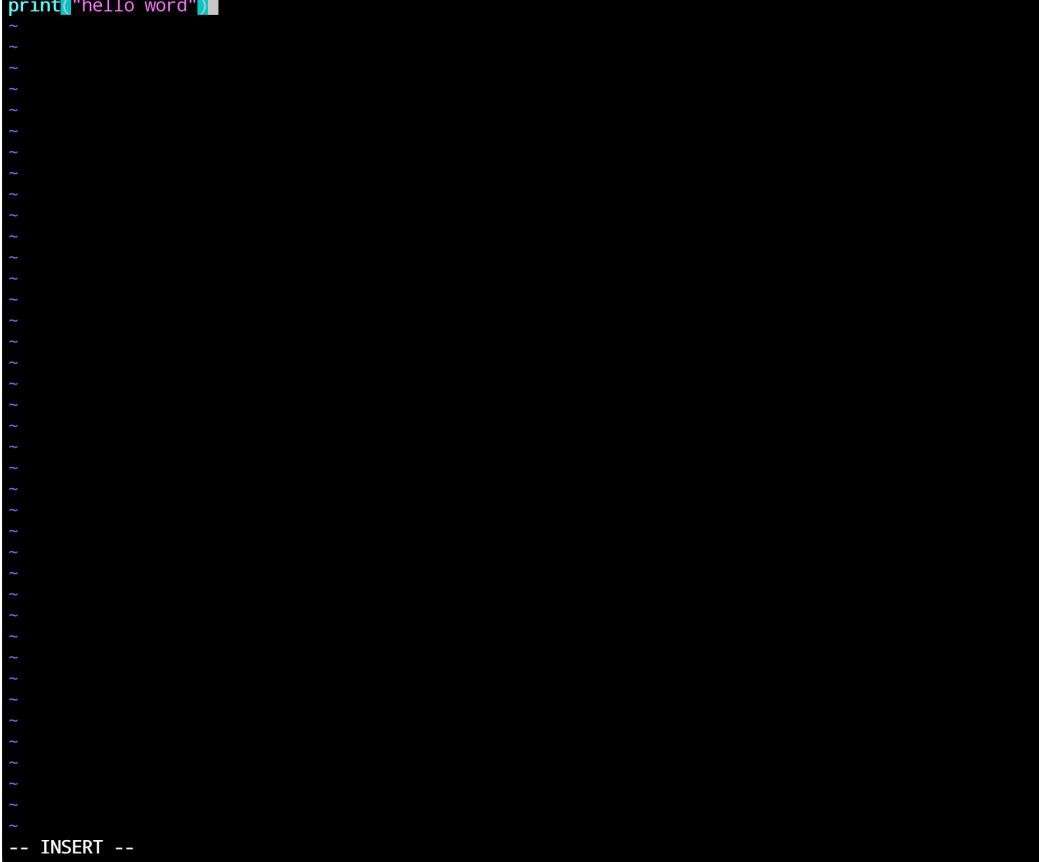
we are now inside the `nlp` folder



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.

```
print("hello word")
```

A screenshot of a terminal window with a black background and white text. On the left side, there is a vertical blue scroll bar. The text area contains a single line of Python code: "print('hello word')". At the bottom of the screen, there is a status bar with the text "-- INSERT --" in white. The overall appearance is that of a standard terminal interface for running code.



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



I'm stuck and cannot escape. It says:

4575



"type :quit<Enter> to quit VIM"



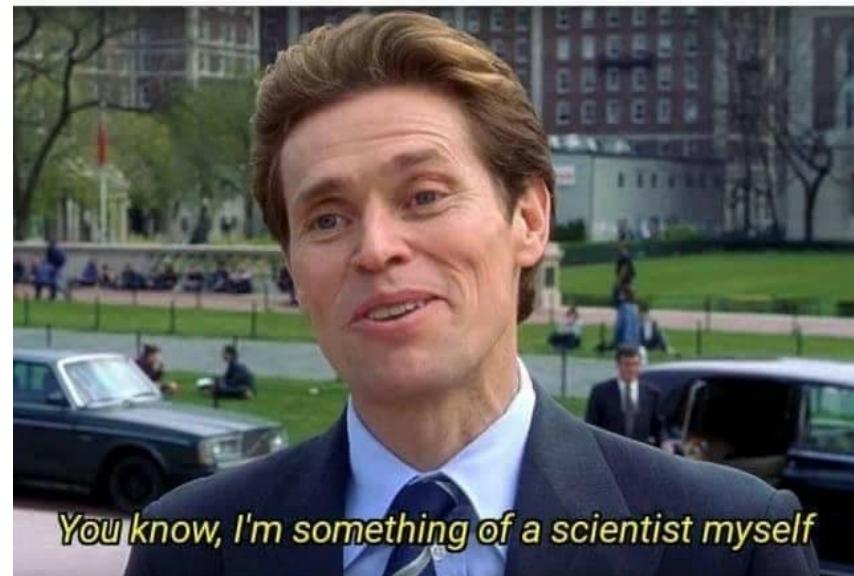
1049



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

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.

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

A screenshot of a terminal window showing a blank black screen. The window title bar at the top has the text "ml-ubuntu20-04-desktop" and the status bar at the bottom shows the command "first_ses0: bash*" and the time "23:57".

<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 a 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-libs -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`

The screenshot shows a list of PyTorch wheels available at https://download.pytorch.org/whl/torch_stable.html. The list includes various combinations of CUDA (cu111, cu110, cu102) and Python (cp37, cp38, cp39) versions, along with their corresponding Linux and Windows builds (amd64, win_amd64). The list is very long, showing every possible combination for each major PyTorch version.

```
cu111/torch-1.10.2%2Bcu111-cp37-cp37m-win_amd64.whl  
cu111/torch-1.10.2%2Bcu111-cp38-cp38-linux_x86_64.whl  
cu111/torch-1.10.2%2Bcu111-cp38-cp38-win_amd64.whl  
cu111/torch-1.10.2%2Bcu111-cp39-cp39-linux_x86_64.whl  
cu111/torch-1.10.2%2Bcu111-cp39-cp39-win_amd64.whl  
cu111/torch-1.8.0%2Bcu111-cp36-cp36m-linux_x86_64.whl  
cu111/torch-1.8.0%2Bcu111-cp36-cp36m-win_amd64.whl  
cu111/torch-1.8.0%2Bcu111-cp37-cp37m-linux_x86_64.whl  
cu111/torch-1.8.0%2Bcu111-cp37-cp37m-win_amd64.whl  
cu111/torch-1.8.0%2Bcu111-cp38-cp38-linux_x86_64.whl  
cu111/torch-1.8.0%2Bcu111-cp38-cp38-win_amd64.whl  
cu111/torch-1.8.0%2Bcu111-cp39-cp39-linux_x86_64.whl  
cu111/torch-1.8.0%2Bcu111-cp39-cp39-win_amd64.whl  
cu111/torch-1.8.1%2Bcu111-cp36-cp36m-linux_x86_64.whl  
cu111/torch-1.8.1%2Bcu111-cp36-cp36m-win_amd64.whl  
cu111/torch-1.8.1%2Bcu111-cp37-cp37m-linux_x86_64.whl  
cu111/torch-1.8.1%2Bcu111-cp37-cp37m-win_amd64.whl  
cu111/torch-1.8.1%2Bcu111-cp38-cp38-linux_x86_64.whl  
cu111/torch-1.8.1%2Bcu111-cp38-cp38-win_amd64.whl  
cu111/torch-1.8.1%2Bcu111-cp39-cp39-linux_x86_64.whl  
cu111/torch-1.8.1%2Bcu111-cp39-cp39-win_amd64.whl  
cu111/torch-1.9.0%2Bcu111-cp36-cp36m-linux_x86_64.whl  
cu111/torch-1.9.0%2Bcu111-cp36-cp36m-win_amd64.whl  
cu111/torch-1.9.0%2Bcu111-cp37-cp37m-linux_x86_64.whl  
cu111/torch-1.9.0%2Bcu111-cp37-cp37m-win_amd64.whl  
cu111/torch-1.9.0%2Bcu111-cp38-cp38-linux_x86_64.whl  
cu111/torch-1.9.0%2Bcu111-cp38-cp38-win_amd64.whl  
cu111/torch-1.9.0%2Bcu111-cp39-cp39-linux_x86_64.whl  
cu111/torch-1.9.0%2Bcu111-cp39-cp39-win_amd64.whl  
cu111/torch-1.9.1%2Bcu111-cp36-cp36m-linux_x86_64.whl  
cu111/torch-1.9.1%2Bcu111-cp36-cp36m-win_amd64.whl  
cu111/torch-1.9.1%2Bcu111-cp37-cp37m-linux_x86_64.whl  
cu111/torch-1.9.1%2Bcu111-cp37-cp37m-win_amd64.whl  
cu111/torch-1.9.1%2Bcu111-cp38-cp38-linux_x86_64.whl  
cu111/torch-1.9.1%2Bcu111-cp38-cp38-win_amd64.whl  
cu111/torch-1.9.1%2Bcu111-cp39-cp39-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
```

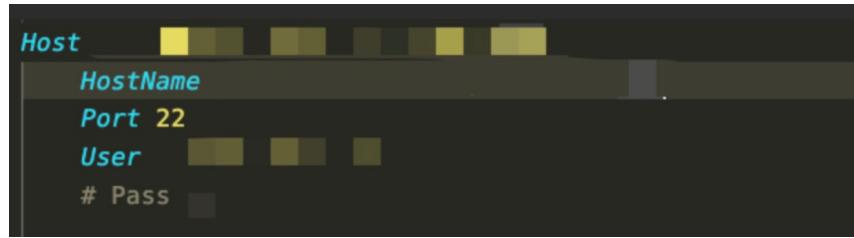
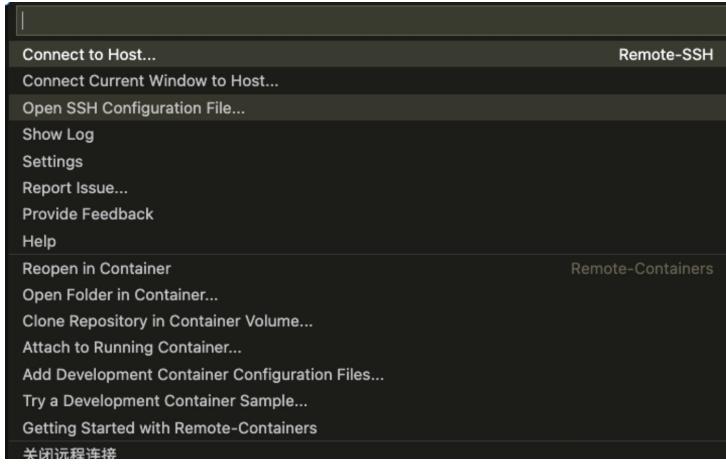


Vscode – remote connection

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



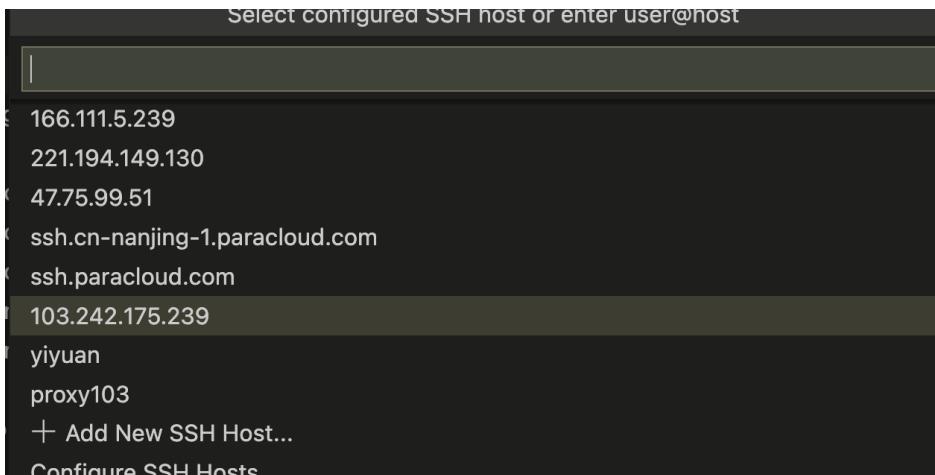
- Select open ssh configuration file, add connection info





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

```
test_parallel > scripts > pytorch_nn.py
1 import torch
2 import math
3
4
5 class Polynomial3(torch.nn.Module):
6     def __init__(self):
7         """
8             In the constructor we instantiate four parameters and assign them as
9             member parameters.
10            """
11            super().__init__()
12            self.a = torch.nn.Parameter(torch.randn(()))
13            self.b = torch.nn.Parameter(torch.randn(()))
14            self.c = torch.nn.Parameter(torch.randn(()))
15            self.d = torch.nn.Parameter(torch.randn(()))
16
17    def forward(self, x):
18        """
19            In the forward function we accept a Tensor of input data and we must return
20            a Tensor of output data. We can use Modules defined in the constructor as
21            well as arbitrary operators on Tensors.
22            """
23
24            return self.a + self.b * x + self.c * x ** 2 + self.d * x ** 3
25
26    def string(self):
27        """
28            Just like any class in Python, you can also define custom method on PyTorch modules
29            """
30
31            return f'y = {self.a.item()} + {self.b.item()} x + {self.c.item()} x^2 + {self.d.item()} x^3'
32
33 # Create Tensors to hold input and outputs.
34 x = torch.linspace(math.pi, math.pi, 2000)
35 y = torch.sin(x)
36
37 # Construct our model by instantiating the class defined above
38 model = Polynomial3()
39
40 # Construct our loss function and an Optimizer. The call to model.parameters()
41 # in the SGD constructor will contain the learnable parameters (defined
42 # with torch.nn.Parameter) which are members of the model.
43 criterion = torch.nn.MSELoss(reduction='sum')
44 optimizer = torch.optim.SGD(model.parameters(), lr=1e-6)
45 for t in range(2000):
46     # Forward pass: compute predicted y by passing x to the model
47     y_pred = model(x)
48
49     # Compute and print loss
50     loss = criterion(y_pred, y)
51     if t % 100 == 99:
52         print(t, loss.item())
53
54     # Zero gradients, perform a backward pass, and update the weights.
55     optimizer.zero_grad()
56     loss.backward()
57     optimizer.step()
58
59 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:
SHA256:6P1c1tTxweKvZLMsg0FPZmIVPxkfkt0dIL3DmfKlauhs 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 AAAAB3NzaC1yc2EAAAQABAAABgQC3G6NS1UJWnuGAj+IumEF0zLGH8mf1QceV5bKdSDJxUgZh9/meanPDUUKfS0pXvn9ReZYf
6Y5h/J1hZsEh2fXV0toAD1wwFwq6NeMbkhMBAaGQdb6RYf2ZXP1HUNHG/ZJjhFiRILCxJ2AhSxu/jhICgRZsDIirIm/4SSTWKFgNkGme+z67P
FszRwpKkrD0931hfJpM4kNLee1fyqFOjBmhGRYxkDMiWhmvZGmbb0Jk070ChJHm1Q05oWRR/UYsNrox0Y0QvLqjEFyk0df7og8744is5m81+
UI3dEBqPF1CnFPExKVfoyeqArEXvzLtUPb/o47/ajNC8FAhy/pnsU8+/Te0ZbMfJZ1Wa74cQ7xQ2k= 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

A screenshot of a web browser displaying the GitHub settings/keys page at github.com/settings/keys. The page shows three SSH keys associated with the user's account. Each key entry includes the key icon, the date it was added, the last used date, and a "Delete" button.

SSH Key	Added	Last Used	Action
[Redacted]	Dec 15, 2020	Within last 3 weeks	Delete
[Redacted]	Jun 21, 2021	Within last 4 weeks	Delete
[Redacted]	Nov 29, 2021	Within last 2 weeks	Delete

The left sidebar shows the user's profile picture and name "DingDing", followed by a list of account settings: Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, **SSH and GPG keys** (which is the active tab), Organizations, Moderation, Code, planning, and automation, and Repositories.



Git Operations

- Git is a code version control tools. Every programmer should learn its basic usage.
- git init git add .
- git commit –m message git clone
- git pull git push
- git checkout git stash
- git pop
- git merge
- .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

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

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

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



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
```



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
```

```
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:
```

\$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.



Bash

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

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

Passing Arguments

```
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.

n>

command > file

Redirecting Outputs

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



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 Linguistics
 - 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