

Gen-AI: Technical and Social

Lecture 02: Introduction to Deep Learning and
Text Classification

Deep Learning: A New Culture

- Data Sharing
- Code Sharing
- Competitive Challenge



Hugging Face



Deep Learning: A New Culture

- Data Sharing
- Code Sharing
- Competitive Challenge

kaggle



Hugging Face



Nice paper



**GitHub
Link**



**Written in
your favourite
framework**




**Runs smoothly on
your system
without error or
dependency issues**





Kaggle



- An ML community you can participate ML competition (19M users)
- You can also share/access open Dataset, Notebook, Models
- A great place to learn from competitions and from each other

 **Active Competitions**

Hotness ▾ 



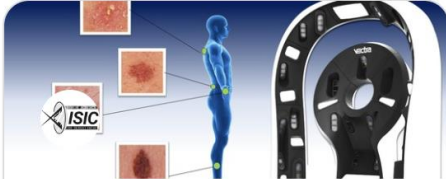
ARC Prize 2024 ⋮

Create an AI capable of solving reason...

Featured · Code Competition

705 Teams

\$1,100,000 3 months to go



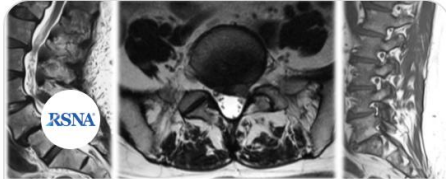
ISIC 2024 - Skin Cancer Detection with 3D-TBP ⋮

Identify cancers among skin lesions cr...

Research · Code Competition

1982 Teams

\$80,000 25 days to go




RSNA 2024 Lumbar Spine Degenerative... ⋮

Classify lumbar spine degenerative co...

Featured · Code Competition

998 Teams

\$50,000 2 months to go



LLM 20 Questions ⋮

Guess the secret word in this cooperat...

Featured · Simulation Competition

826 Teams

\$50,000 a day to go

Hugging Face




- An AI community where people share Models, Datasets, AI apps (800K models, 200K datasets, etc..)
- Their `transformers` package is widely used for Gen-AI

Models 825,985

 Filter by name

Full-text search

↑↓ Sort: Most downloads

 MIT/ast-finetuned-audioset-10-10-0.4593

 Audio Classification • Updated Sep 6, 2023 • ⬇ 269M • ❤ 221



 microsoft/resnet-50

 Image Classification • Updated Feb 13 • ⬇ 71.7M • ❤ 259

 facebook/fasttext-language-identification

 Text Classification • Updated Jun 9, 2023 • ⬇ 52.7M • ❤ 166

 sentence-transformers/all-MiniLM-L6-v2

 Sentence Similarity • Updated May 29 • ⬇ 47.2M • ⚡ • ❤ 2.15k

GitHub



- A place where people share Code (Repo)
- A great place to learn coding

Let's build from here

The complete developer platform to build, scale, and deliver secure software.

100+ million

Developers

4+ million

Organizations

420+ million

Repositories

90%

Fortune 100



pytorch



Fork 21.8k



Star 81.2k



76,997 Commits

Contributors 3,433



transformers



Fork 25.9k



Star 130k



16,585 Commits

Contributors 2,664

A Deep Learning Task

- Given a dataset $\{(x_i, y_i)\}_{i=1}^N$, learn a function to minimize the prediction loss

$$\min_{\theta} \sum_{i=1}^N L(f(x_i; \theta), y_i)$$

- We learn θ from a training dataset, and evaluate its performance on a separate validation/test dataset

A Deep Learning Task

- Given a dataset $\{(x_i, y_i)\}_{i=1}^N$, learn a function to minimize the prediction loss

$$\min_{\theta} \sum_{i=1}^N L(f(x_i; \theta), y_i)$$

- We learn θ from a training dataset, and evaluate its performance on a separate validation/test dataset
- L is the objective function/loss function: e.g., MSE, Cross-entropy etc
- f is typically an artificial neural network and the optimization is typically done by first-order methods

Hands-On DL Example

When I cannot create, I do not understand --- Feynman

Predicting Heart Disease

Using a dataset of patients made available by the Cleveland Clinic, we will build our first DL model to predict if a patient has been diagnosed with heart disease from demographics and bio-markers

Column	Description	Feature Type
Age	Age in years	Numerical
Sex	(1 = male; 0 = female)	Categorical
CP	Chest pain type (0, 1, 2, 3, 4)	Categorical
Trestbpd	Resting blood pressure (in mm Hg on admission)	Numerical
Chol	Serum cholesterol in mg/dl	Numerical
FBS	fasting blood sugar in 120 mg/dl (1 = true; 0 = false)	Categorical
RestECG	Resting electrocardiogram results (0, 1, 2)	Categorical
Thalach	Maximum heart rate achieved	Numerical
Exang	Exercise induced angina (1 = yes; 0 = no)	Categorical
Oldpeak	ST depression induced by exercise relative to rest	Numerical
Slope	Slope of the peak exercise ST segment	Numerical
CA	Number of major vessels (0-3) colored by fluoroscopy	Both numerical & categorical
Thal	3 = normal; 6 = fixed defect; 7 = reversible defect	Categorical
Target	Diagnosis of heart disease (1 = true; 0 = false)	Target

What we want to predict

Let's design our NN

- We design i.e., “lay out” the network
 - Choose *the number of hidden layers* and *the number of ‘neurons’ in each layer*
 - Pick the *right output layer* based on the type of the output

Let's design our NN

- We design i.e., “lay out” the network
 - Choose *the number of hidden layers* and *the number of ‘neurons’ in each layer*
 - Pick the *right output layer* based on the type of the output

1 hidden layer (16 neurons) + Sigmoid output

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

Let's visualize this NN

Input
Layer

x_1

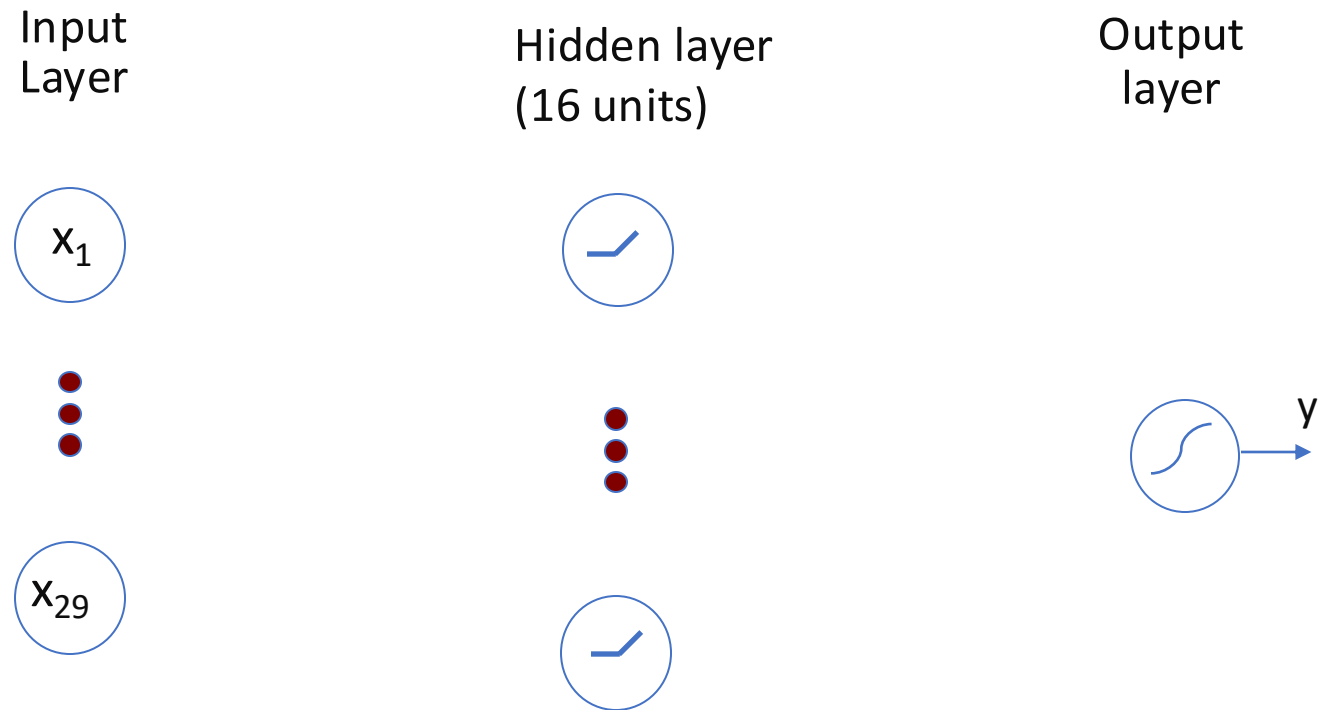
⋮

x_{29}

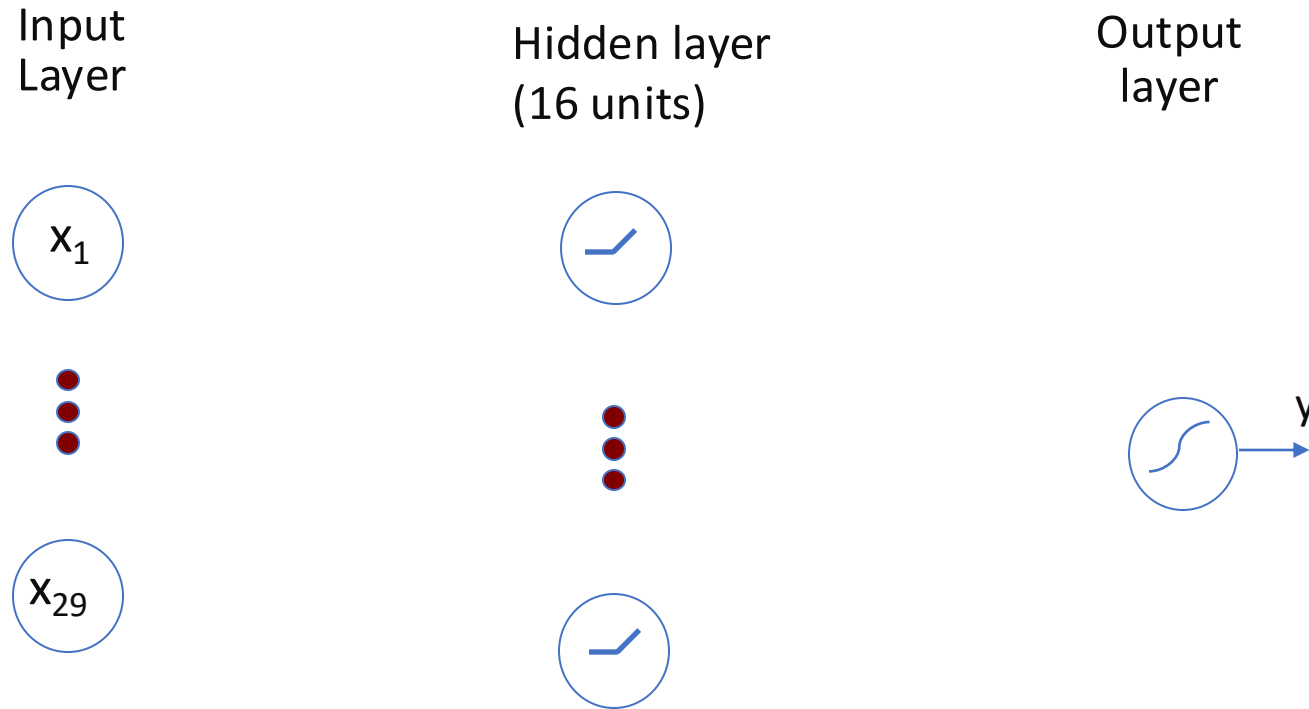
Column	Description	Feature Type
Age	Age in years	Numerical
Sex	(1 = male; 0 = female)	Categorical
CP	Chest pain type (0, 1, 2, 3, 4)	Categorical
Trestbpd	Resting blood pressure (in mm Hg on admission)	Numerical
Chol	Serum cholesterol in mg/dl	Numerical
FBS	fasting blood sugar in 120 mg/dl (1 = true; 0 = false)	Categorical
RestECG	Resting electrocardiogram results (0, 1, 2)	Categorical
Thalach	Maximum heart rate achieved	Numerical
Exang	Exercise induced angina (1 = yes; 0 = no)	Categorical
Oldpeak	ST depression induced by exercise relative to rest	Numerical
Slope	Slope of the peak exercise ST segment	Numerical
CA	Number of major vessels (0-3) colored by fluoroscopy	Both numerical & categorical
Thal	3 = normal; 6 = fixed defect; 7 = reversible defect	Categorical
Target	Diagnosis of heart disease (1 = true; 0 = false)	Target

There are only 13 input variables but some of them are categorical so we one-hot-encode them, resulting in 29 inputs (details in colab).

Let's visualize this NN

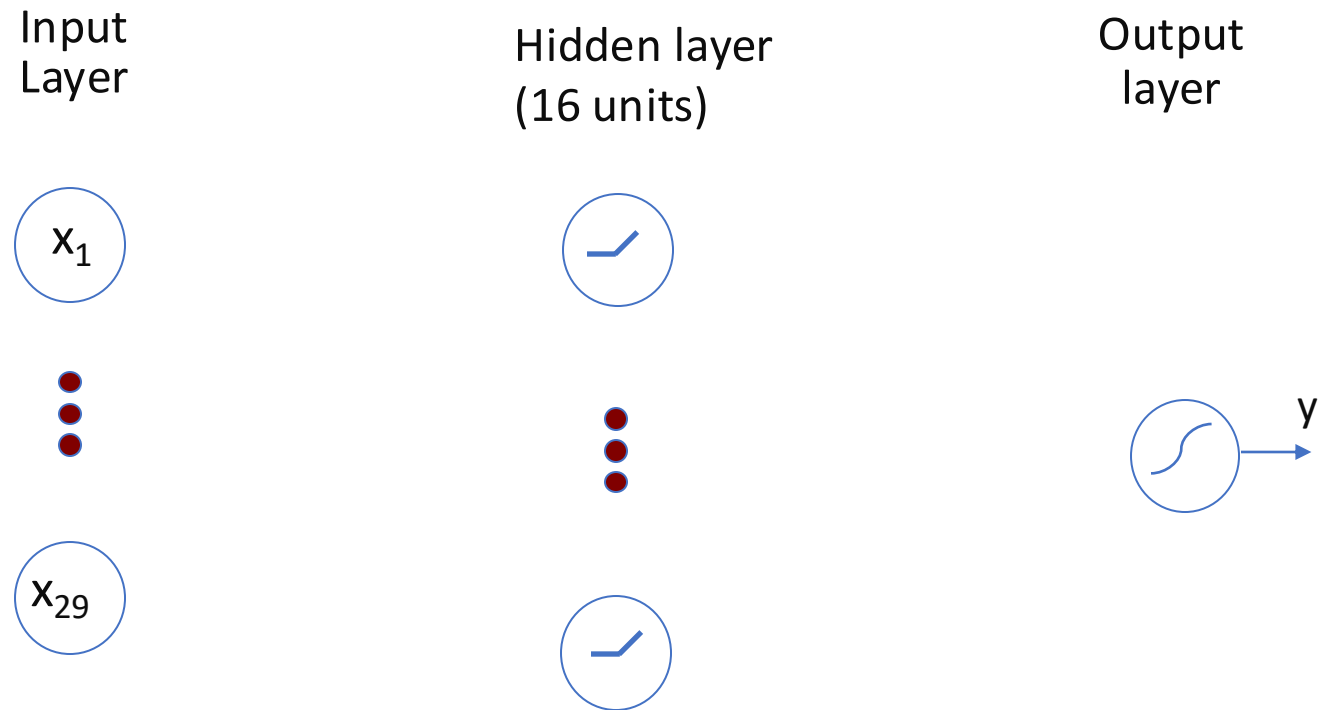


How to construct this NN in PyTorch



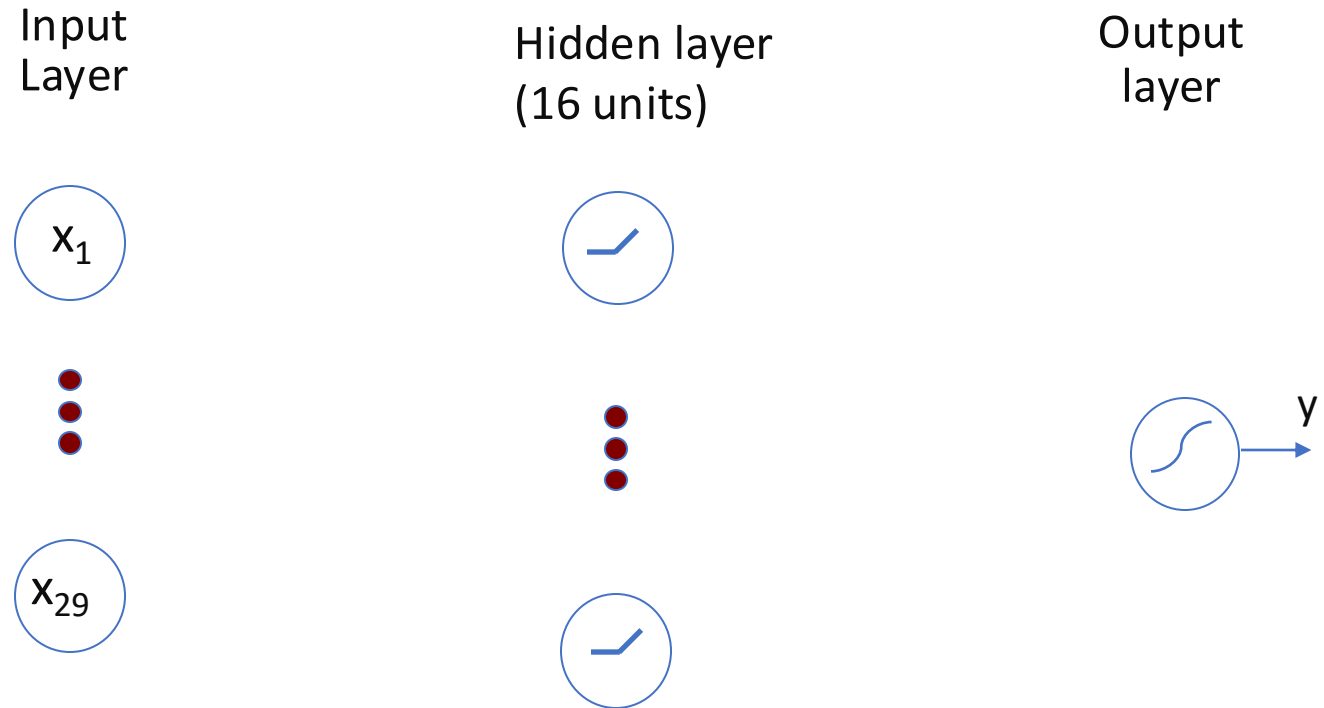
```
out = nn.Linear(29, 16)(input)
out = nn.ReLU()(out)
out = nn.Linear(16, 1)(out)
out = nn.Sigmoid()(out)
```

Let's visualize this NN



How many parameters (i.e., weights and biases) does this network have?

Let's visualize this NN



How many parameters (i.e., weights and biases) does this network have?

$$(29 \cdot 16 + 16) + (16 + 1) = 497$$

What should be the right loss function

- How to measure the difference of two distributions p (target) and q

$$D_{KL}(p||q) = - \sum_i p_i \log(q_i) + \sum_i p_i \log(p_i)$$

Cross-entropy $H(p, q)$

Entropy of p

What should be the right loss function

- How to measure the difference of two distributions p (target) and q

$$\arg \min_{q \in \mathcal{Q}} D_{KL}(p||q) = - \sum_i p_i \log(q_i) + \sum_i p_i \log(p_i)$$

Cross-entropy $H(p, q)$

Entropy of p

What should be the right loss function

- How to measure the difference of two distributions p (target) and q

$$\arg \min_{q \in \mathcal{Q}} \quad H(p, q) = - \sum_i p_i \log(q_i)$$

Cross-entropy $H(p, q)$

What should be the right loss function

- How to measure the difference of two distributions p (target) and q

$$\arg \min_{q \in \mathcal{Q}} H(p, q) = - \sum_i p_i \log(q_i)$$

Cross-entropy $H(p, q)$

- For binary class, if $p_0 = 1$, then $H(p, q) = -\log(q_0)$ otherwise $-\log(q_1)$

What should be the right loss function

- How to measure the difference of two distributions p (target) and q

$$\arg \min_{q \in \mathcal{Q}} H(p, q) = - \sum_i p_i \log(q_i)$$

Cross-entropy $H(p, q)$

- For binary class, if $p_0 = 1$, then $H(p, q) = -\log(q_0)$ otherwise $-\log(q_1)$
- This is called Binary Cross Entropy Loss (BCE loss)

```
torch.nn.BCELoss()(prediction, target)
```

Let's go over the Colab together

- Notebook link:

[https://colab.research.google.com/drive/1PNF64MQ1v1ZQn7cqYs5cB
SKKdeE-tVZ5?usp=sharing](https://colab.research.google.com/drive/1PNF64MQ1v1ZQn7cqYs5cBSKKdeE-tVZ5?usp=sharing)

- You can learn more about PyTorch from

- https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Text Classification

Where is the wisdom we have lost in knowledge? Where is the knowledge we have lost in information?

--- T.S. Eliot (1888-1965, poet)

Text Classification

- Practically, categorizing texts have a wide range of applications
 - Content moderation
 - Spam detection
 - Sentiment analysis
 - Intent detection
 - Automatic keyword generation
 -
- Today, we will discuss how people have traditionally solved text classification and how it has evolved over time

A Text Classification Task: Stress Detection

- Given a piece of text, detect the stress level of the author who wrote the text

A Text Classification Task: Stress Detection^[1]

- 3.5K text segments
 - Collected from 5 subreddits post: r/abuse, r/anxiety, r/financial, r/PTSD, r/social
 - Labeled as “Stress” or “Not Stress” or “Cannot Tell” by at least 5 Amazon Mechanical Turks
 - Take majority as the ground truth (with “Cannot Tell” majority posts removed)

[1] Dreddit: A Reddit Dataset for Stress Analysis in Social Media (Turcan & McKeown, Louhi 2019)

A Text Classification Task: Stress Detection^[1]

Text	Label	Ave. Agreed
...These past couple of months have been the worst. My anxiety has gotten so bad it's effecting my sleep and relationship... I don't have money to see a therapist either..	Stress	100%
Sometimes it really sucks whenever I don't have anything else to give to others. But now that it's winter.. I gave away 4 sleeping bags that we no longer use...	Not Stress	80%

[1] Dreddit: A Reddit Dataset for Stress Analysis in Social Media (Turcan & McKeown, Louhi 2019)

First step: Tokenization

- Tokenization: given a text string, we need to get a sequence of integers

First step: Tokenization

- Tokenization: given a text string, we need to get a sequence of integers

What's your thought?

First step: Tokenization

- Tokenization: given a text string, we need to get a sequence of integers
 - 1) map each word to an integer?
 - Too many words (many are rarely used, not work for Chinese and others)

First step: Tokenization

- Tokenization: given a text string, we need to get a sequence of integers
 - 1) map each word to an integer?
 - Too many words (many are rarely used, not work for Chinese and others)
 - 2) map each character to an integer?
 - The sequence length increases significantly, loses semantic meaning
 - E.g., “triangulation” needs to 13 integers

First step: Tokenization

- Tokenization: given a text string, we need to get a sequence of integers
 - 1) map each word to an integer?
 - Too many words (many are rarely used, not work for Chinese and others)
 - 2) map each character to an integer?
 - The sequence length increases significantly, loses semantic meaning
 - E.g., “triangulation” needs to 13 integers
 - 3) sub-word tokenization?
 - Yes! That’s what GPT is using
 - “triangulation” -> [“tri”, “ang”, “ulation”] (<https://platform.openai.com/tokenizer>, one token corresponds to about 4 characters or $\frac{3}{4}$ of a word)

First step: Tokenization

- How to find sub-word?

First step: Tokenization

- How to find sub-word? Byte-Pair Encoding (BPE)
- Given a large corpus ['ababababacdda', 'abbccbbbaabd', '....']
- 1. Initialize the vocabulary of tokens with characters ['a', 'b', 'c', 'd']

First step: Tokenization

- How to find sub-word? Byte-Pair Encoding (BPE)
- Given a large corpus ['ababababacdda', 'abbccbbbaabd', '....']
- 1. Initialize the vocabulary of tokens with characters ['a', 'b', 'c', 'd']
- 2. Find the pair of tokens that occurs most frequently in the corpus, say, 'ab', then insert 'ab' to the vocabulary -> ['a', 'b', 'c', 'd', 'ab']

First step: Tokenization

- How to find sub-word? Byte-Pair Encoding (BPE)
- Given a large corpus ['ababababacdda', 'abbccbbbaabd', '....']
- 1. Initialize the vocabulary of tokens with characters ['a', 'b', 'c', 'd']
- 2. Find the pair of tokens that occurs most frequently in the corpus, say, 'ab', then insert 'ab' to the vocabulary -> ['a', 'b', 'c', 'd', 'ab']
- 3. Repeat this process until the vocabulary size reaches a limit

First step: Tokenization

- How to find sub-word? Byte-Pair Encoding (BPE)
- Given a large corpus ['ababababacdda', 'abbccbbbaabd', '....']
- 1. Initialize the vocabulary of tokens with characters ['a', 'b', 'c', 'd']
- 2. Find the pair of tokens that occurs most frequently in the corpus, say, 'ab', then insert 'ab' to the vocabulary -> ['a', 'b', 'c', 'd', 'ab']
- 3. Repeat this process until the vocabulary size reaches a limit

```
1 from transformers import AutoTokenizer
2
3 tokenizer = AutoTokenizer.from_pretrained('GPT2')
4 tokenizer(['Go go power ranger'])
```

```
{'input_ids': [[5247, 467, 1176, 43570]], 'attention_mask': [[1, 1, 1, 1]]}
```

The first model to build

- What's the simplest model you can think of?

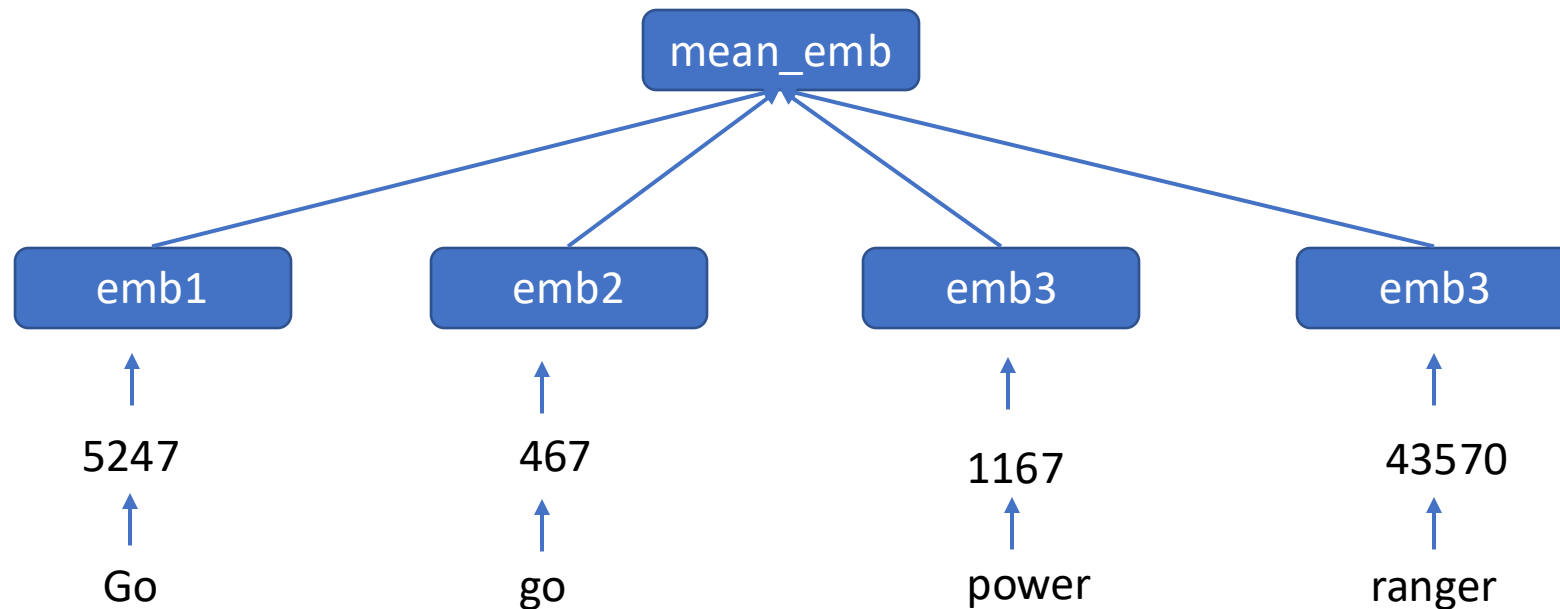
The first model to build

- What's the simplest model you can think of?
- Bag of words: each token has an embedding, we take the mean of all embeddings



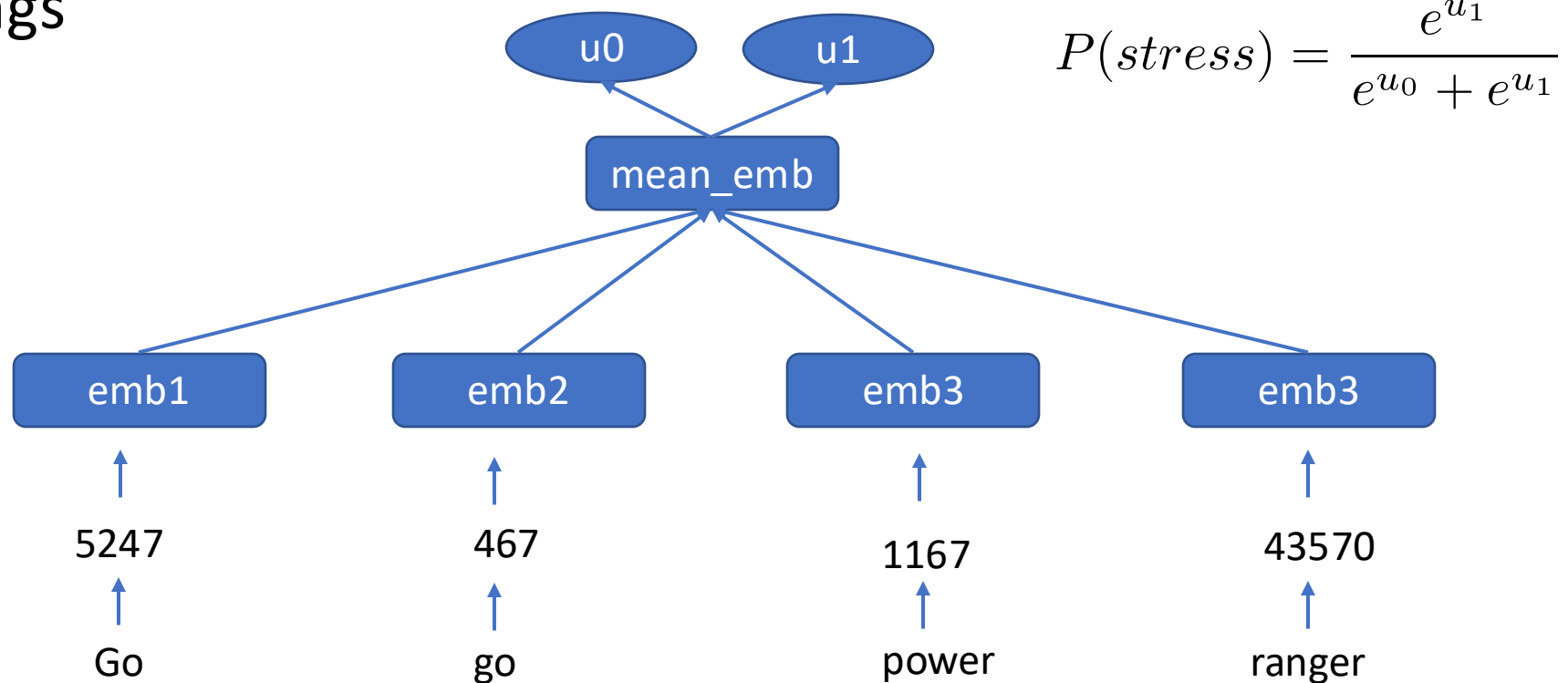
The first model to build

- What's the simplest model you can think of?
- Bag of words: each token has an embedding, we take the mean of all embeddings



The first model to build

- What's the simplest model you can think of?
- Bag of words: each token has an embedding, we take the mean of all embeddings



A few lines of code

```
import torch
import torch.nn as nn

class EmbeddingModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim, num_classes):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.fc = nn.Linear(embedding_dim, num_classes) # Pooling is done before this
        self.relu = nn.ReLU()

    def forward(self, inputs):
        output = self.embedding(inputs)
        pooled_output = torch.mean(output, dim=1) # Mean pooling
        pooled_output = self.relu(pooled_output)
        logits = self.fc(pooled_output)
        return logits

model = EmbeddingModel(tokenizer.vocab_size, 128, 2)
```

Word Embeddings

- How can we learn meaningful word embeddings?
 - Such that related words should have closer distance?

Word Embeddings

- How can we learn meaningful word embeddings?
 - Such that related words should have closer distance?
 - Maybe the pair of words that occur more frequently in a text should have closer embedding?

Word Embeddings

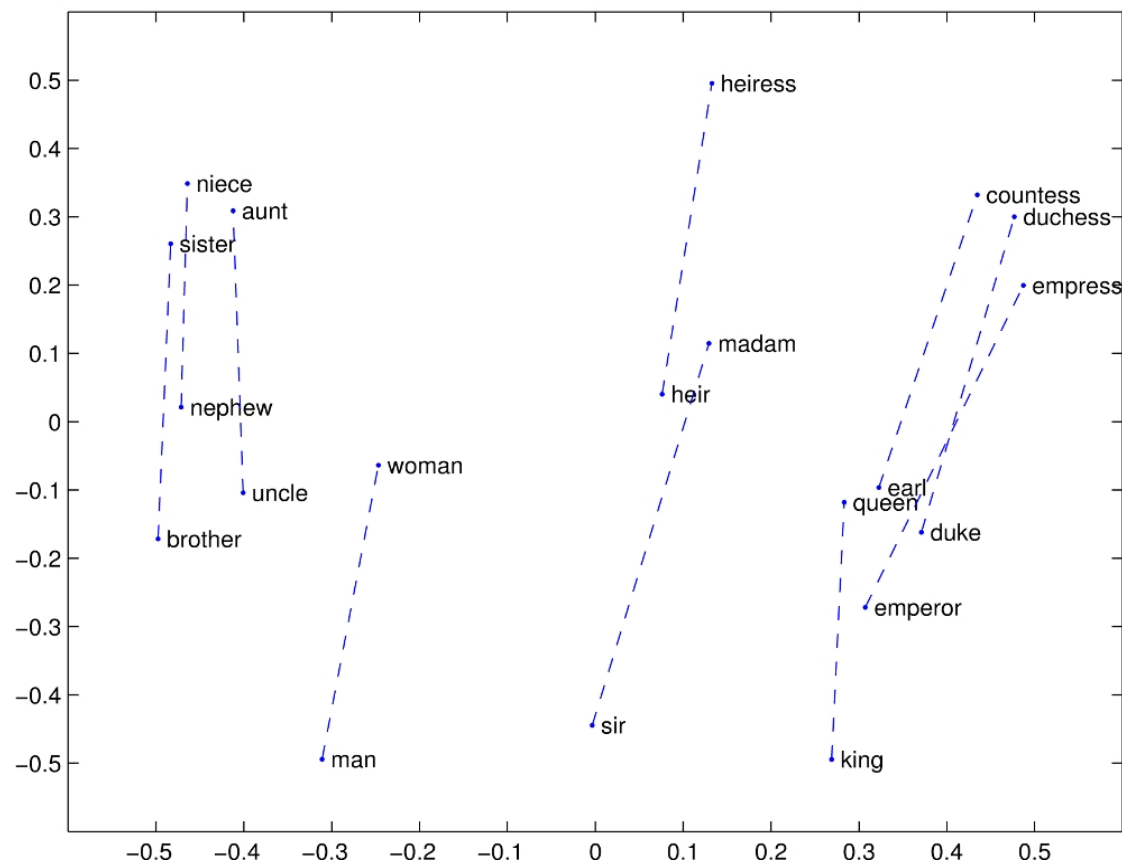
- How can we learn meaningful word embeddings?
 - Such that related words should have closer distance?
 - Maybe the pair of words that occur more frequently in a text should have closer embedding?
- GloVe exactly did that
 - Trained on 1-2GB data

[PDF] **Glove: Global vectors for word representation**

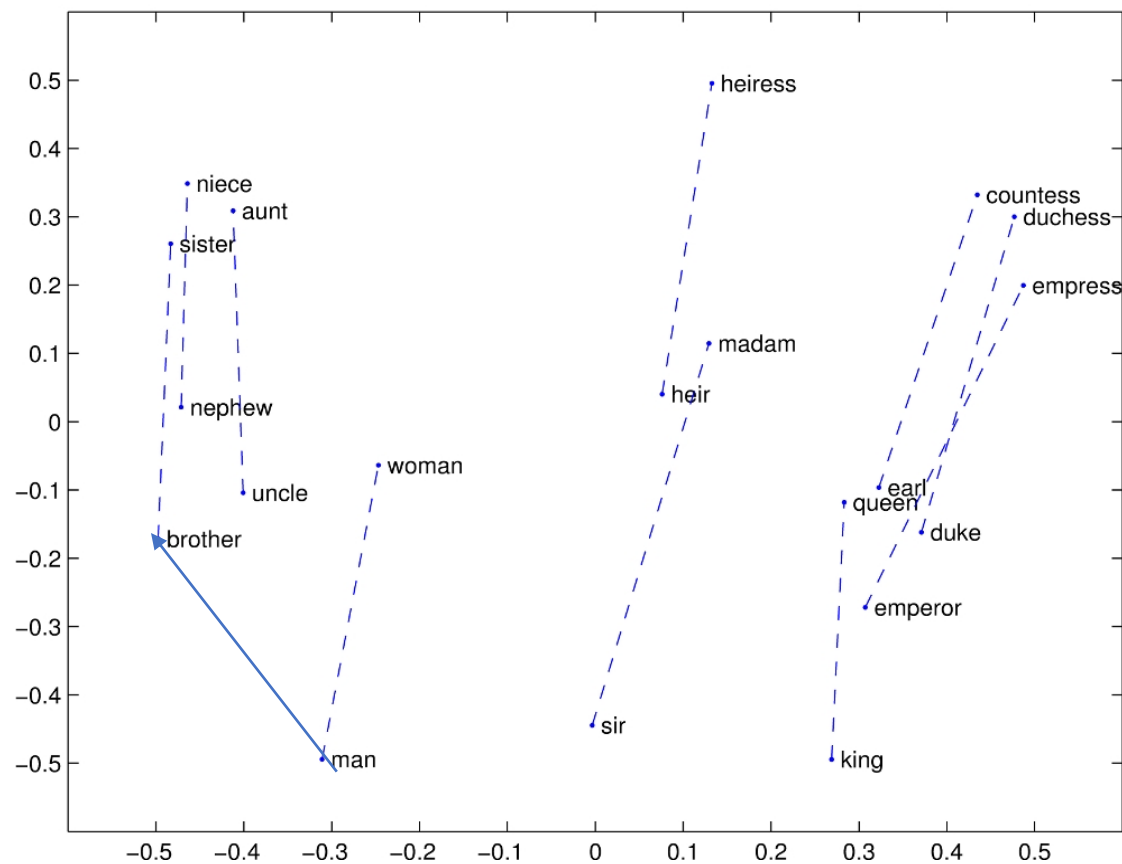
J Pennington, R Socher, CD Manning - Proceedings of the 2014 conference on ..., 2014

☆  Cite Cited by 42249 Related articles All 27 versions

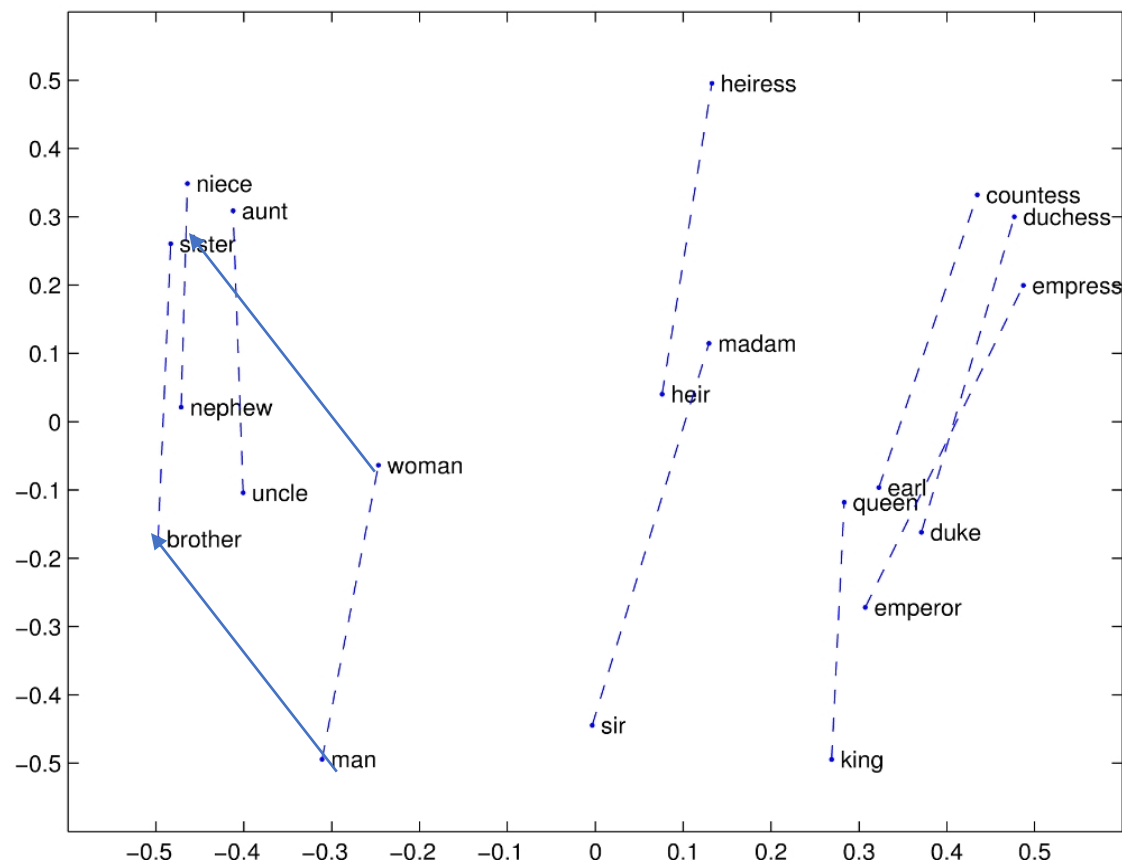
The geometry of word embeddings



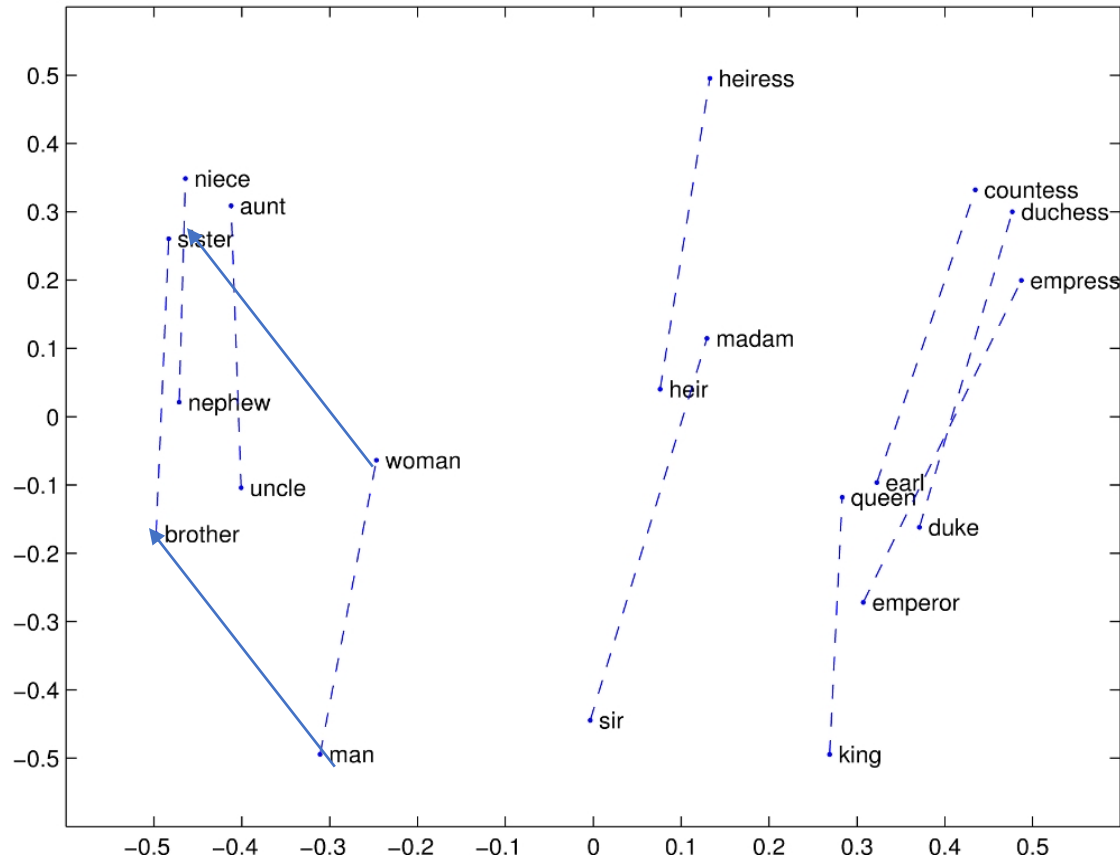
The geometry of word embeddings



The geometry of word embeddings



The geometry of word embeddings



$(\text{brother} - \text{man}) + \text{woman}$
= sister

Issues of Word Embeddings?

Issues of Word Embeddings?

- 1. Does not take the surrounding contexts into account
- 2. Does not take care of the order of the words

Learning Sentence Embeddings

- Transformer fixes all the issues:
 - 1. Does not take the surrounding contexts into account
 - 2. Does not take care of the order of the words

Learning Sentence Embeddings

- Transformer fixes all the issues:
 - 1. Does not take the surrounding contexts into account
 - 2. Does not take care of the order of the words
- We will cover the details of Transformer in the next lecture

Learning Sentence Embeddings

- Transformer fixes all the issues:
 - 1. Does not take the surrounding contexts into account
 - 2. Does not take care of the order of the words
- We will cover the details of Transformer in the next lecture
- This lecture we will see how to directly use the sentence embedding provide by Bert directly

Learning Sentence Embeddings

- Notebook link:
<https://colab.research.google.com/drive/15TaJkbClT0JF89ZsYplgj9mjki4KLqkm?usp=sharing>

How about prompting LLM for classification?

- Let's do a small task together (see 20 examples from the LLM section)

Instructions: Below you are given snippets of text from social media. Please decide whether the person who wrote each snippet is, overall, stressed about what they're talking about.

If you believe the writer is stressed and has ****an overall negative attitude**** about it, output Yes.

If you believe the writer is not stressed, expresses stress but does not have an overall negative attitude about it, or has a negative attitude but you don't consider it stress (e.g., angry but not stressed), output No.

Remember to decide based on the feelings the writer is expressing, NOT whether you think the situation is or should be stressful.

How about prompting LLM for classification?

The true labels of the provided texts 0 indicates No Stress, 1 indicates Stress What is your accuracy?	0	0
	1	0
	2	1
	3	1
	4	0
	5	0
	6	1
	7	0
	8	1
	9	0
	10	1
	11	1
	12	0
	13	1
	14	0
	15	1
	16	1
	17	0
	18	1
	19	0

How about prompting LLM for classification?

- Let's test our prompt directly on GPT4o
 - 72% accuracy, not bad, remember we are zero-shot!

How about prompting LLM for classification?

- Let's test our prompt directly on GPT4o
 - 72% accuracy, not bad, remember we are zero-shot!
- How to improve?
 - Chain of thought provides 80%, close to human-level!

```
Your response should be a JSON object with the following format:  
{  
  "analysis": "Your analysis of the text",  
  "stress": "Yes" or "No"  
}
```

How about prompting LLM for classification?

- Let's test our prompt directly on GPT4o
 - 72% accuracy, not bad, remember we are zero-shot!
- How to improve?
 - Chain of thought provides 80%, close to human-level!
- How to get a probability instead of a binary?

How about prompting LLM for classification?

- Let's test our prompt directly on GPT4o
 - 72% accuracy, not bad, remember we are zero-shot!
- How to improve?
 - Chain of thought provides 80%, close to human-level!
- How to get a probability instead of a binary?
 - 1. Directly ask a probability
 - 2. Get top-k token probability using API

Recap: Text Classification

- Approaches:
 - Learn a model from scratch
 - Fine-tune a model
 - Zero-shot LLM

Homework: A Kaggle Exercise For LLM Classification Fine-tuning

<https://www.kaggle.com/competitions/llm-classification-finetuning/data?select=train.csv>

▲ prompt	▲ response_a	▲ response_b	# winner_model_a	# winner_model_b
51734 unique values	56566 unique values	56609 unique values	 01	 01
["Is it morally right to try to have a certain percentage of females on managerial positions?", "OK, ...	["The question of whether it is morally right to aim for a certain percentage of females in manageri...	["As an AI, I don't have personal beliefs or opinions. However, I can tell you that the question of ...	1	0

gpt-4-1106-preview gpt-4-0613

Next Week

Speaker from Anthropic