PHIL 7001: Fundamentals of AI, Data, and Algorithms

Week 11
Foundation Models and Generative AI

Boris Babic,
HKU 100 Associate Professor of Data Science, Law and Philosophy

Learning goals

- The transition from Natural Language Processing (NLP) to Large Language Models (LLMs)
- Basic concepts of LLMs
- How do LLMs work
- Real-world applications of LLMs.
- Social impact and relevance of LLMs, including ethical considerations, biases, legal implications, and future problems.

What is a language model?

A language model is a machine learning model that aims to predict and generate plausible language. Autocomplete is a language model, for example.
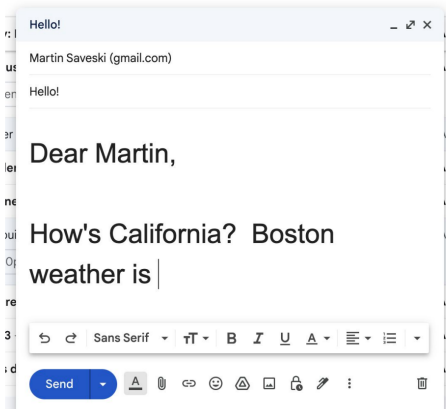
What is a language model?

In this case, the model will generate a probability distribution over all the sequences of words that might be spoken or written (in some language, in some context).

Sentence	Probability
Aardvarks ate apples	0.00000000241
...	...
Boston weather is callous	0.0000000121
Boston weather is cold	0.0000234
Boston weather is cork	0.00000000291
Boston weather is crane	0.00000000185
Boston weather is crazy	0.00000322
Boston weather is furious	0.00000000112
Boston weather is frigid	0.0000321
...	...
Zyzzyx zork zaphod	0.00000000112

Week 9

Boris
Babic,
HKU

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

LLM -
Attention
Heads

LLM -
MLPs

LLM - Un-
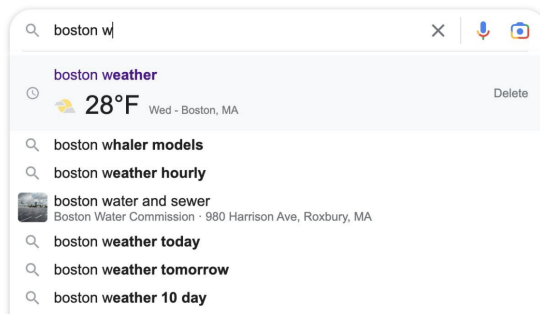embedding

Wrap-up

Boston weather is...

Given that the first three words are 'Boston weather is', the model predicted that the next word should most likely be 'cold'.

w	P(w | Boston weather is)
cold	0.4
frigid	0.23
terrible	0.12
great	0.02
fantastic	0.01
miserable	0.008
warm	0.005
hot	0.002

"Auto-complete" interfaces

Classical applications of LMs: as translators

Language Translation

Classical applications of LMs: as generators

ChatGPT

What are you?

I'm a large language model trained by OpenAI. I'm a form of artificial intelligence that has been designed to process and generate human-like language.

Are you human?

I'm not a human and I don't have the ability to think or feel in the same way that a person does.

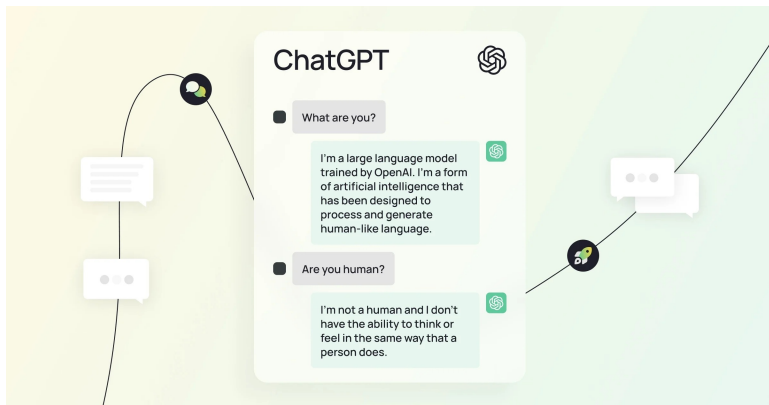## Some Recaps on NLP
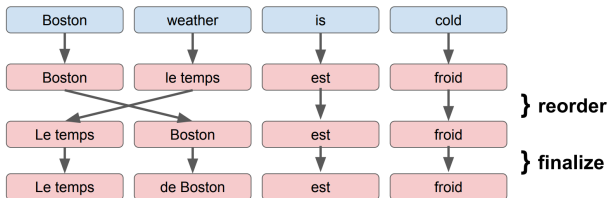
- Natural Language Processing (NLP) is a branch of artificial intelligence (AI) that enables computers to understand, interpret, and generate human language.

- There are different ways to approach an understanding of language, such as approaches based on synonyms, antonyms, and similarity.

- A wide range of practical applications, including chatbots, language translation, sentiment analysis, information retrieval, and text summarization.

- But faces challenges due to the complexity and ambiguity of human language, including variations in grammar, context, and cultural nuances.

Week 9

Boris
Babic,
HKU

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

LLM -
Attention
Heads

LLM -
MLPs

LLM - Un-
embedding

Wrap-up

NLP to LLM

- The rise of deep learning in NLP: Beyond individual words to entire sentences.
    - Traditional NLP techniques often relied on statistical methods to process and analyze language.
    - However, these networks were still limited in their ability to handle very long dependencies in text.
- Large Language Models (LLM): From rule-based systems to probabilistic models and neural networks.
    - We've seen RNN and Seq2Seq as examples.
- The transition from NLP to LLMs: a shift from smaller-scale models to highly complex models with a vast number of parameters capable of generalizing over a large corpus of text.
- Built upon NLP techniques, LLMs can generate coherent and contextually relevant text over longer passages, answer questions, summarize text, and even generate content in response to prompts.

Week 9

Boris
Babic,
HKU

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

LLM -
Attention
Heads

LLM -
MLPs

LLM - Un-
embedding

Wrap-up

Large Language Models: An Introduction

- The Transformer architecture, introduced in the paper "Attention is All You Need" by Vaswani et al., presented a significant shift from early NLP models.

- It introduced the innovated **self-attention mechanisms**, enabling the model to understand the context and relationships between words or subwords in a sentence.

- **Large Language Models (LLMs) are a class of deep learning models specifically designed to understand, generate, and work with human language at a large scale. They have a large number of parameters (often in the billions) and are trained on vast corpora of text data.**

- The self-attention mechanism of Transformers enables LLMs to handle the complexities and subtleties of human language effectively.

- LLMs scale up the Transformer concept by adding more layers (depth) and more parameters, allowing them to capture a broader range of linguistic patterns and generalize better across different language tasks.

- GPT-3 by OpenAI, with 175 billion parameters, is one of the most well-known examples of an LLM.

General Steps in LLM Processing

- **Embedding**: Tokens (words or subwords) are mapped to vectors, capturing both the token's identity and its position within the sequence.
- **Attention Mechanism**: Through self-attention, the model determines the importance of all other tokens in the context of each token, figuring out how the words (now encoded as vectors) relate to one another.
  - It determines what other words in the text to pay attention to and how much attention to pay.
  - It then copies information connected with those words over to the target word. (This was the big innovation involved with transformers.)
- **Multi-Layer Perceptrons (MLPs)**: Each token's vector will then be independently processed through MLPs, which apply non-linear transformations to capture more complex relationships between tokens.
- **Unembedding**: The model translates the final vectors back into token probabilities, ready for text generation.

Embedding

- In this step, words or tokens will be translated from our language into a numerical format that a machine learning model can process. This numerical format is typically a vector, which is essentially a list of numbers.

- At this point, there's no information encoded about surrounding words.

- One-hot encoding is a simple example, where each word is represented as a vector with all zeros except for a one in the position corresponding to the word in the vocabulary.

- To illustrate, let's use a toy model with a six-word vocabulary: "Amy," "Arnett," "Be," "Saw," "Will," "Yesterday." Using one-hot encoding:

$$\text{Amy}: \quad [1, 0, 0, 0, 0, 0]$$
$$\text{Saw}: \quad [0, 0, 0, 1, 0, 0]$$
$$\text{Will}: \quad [0, 0, 0, 0, 1, 0]$$

However, this encoding tells us nothing about the relationship between these words.

Embedding

In practice, we enhance these vectors with more information. We might come up with a list of traits we could associate with each word in our vocab and give it a score. One trait might be something like 'fruitiness', and both 'apples' and 'oranges' would get a high score for this trait.

Features:	is a word	is a noun	is singular	related to food	...
king	1	1	1	0	...
queens	1	1	0	0	...
eating	1	0	0	1	...

vector representation of words

From the vectors, we learn that the word 'queens' is a word and a noun, and 'eating' is a word and is related to food.

Embedding

Think of it like this. You have an Excel file, with each row corresponding to one word in your vocab. The columns are the traits [fruitiness, male-genderedness, edibility, footwear-iness, farm-animalness, redness,...]. When two words get similar scores for a column, you know they bear some kind of similarity. Likewise, when they get opposite scores, you know they have some dissimilarity.

Embedding

- The embeddings can predict unseen words in a text based on their similarity to other words. For example, if our model has learned the concept of "juice" associated with "pineapple" and "grape," it can predict "juice" when it encounters "orange."

- In addition to these features, we also encode the position of words in the sentence. In our previous example, "Amy" is the first word, "saw" is the second, and "Will" is the third. This positional information is crucial for the model to understand the order and syntax of the sentence.

- While our toy model uses a simple four-dimensional space for illustration, real-world models like GPT-3 use much higher-dimensional spaces, with GPT-3 Small using 768 dimensions.

Attention Heads

- We've already encoded each token in the sentence into a unique vector.

- However, a model might treat the words as a series of isolated tokens, not capturing their intricate relationships that are crucial for predicting what comes next.

- Attention heads are components that allow the model to understand the context in which words appear. Their role is to **determine the context and relevance of each word in relation to others within a text**.

- Suppose you feed the model "Right now, John and Jessica" but don't have any way to move information from one word to the next. Without the attention mechanism, the model won't realize that a present tense verb is much more likely to come next than a past tense one, nor that a plural verb is much more likely than a singular one.

- So what attention will ultimately be able to do is allow the embedding for 'Jessica' to receive information from the words preceding it so that, 'are' is much more likely than 'was' based on the initial embedding.

Attention Heads: Two Separate Algorithms

A given attention head does two main things.

- One part is figuring out which previous words to attend to, which we'll call the attention side of the algorithm.

- The other part is figuring out what information to move from a previous word (the source) to the current word (the destination) given that the source is attended to, which we'll call the value side of the algorithm.

The Attention Side

The attention side of an attention head is concerned with determining which parts of the input sequence are relevant to each other. This relevance is quantified in terms of attention scores, which are essentially weights indicating how much each token should consider other tokens when updating its own representation.

Boris
Babic,
HKU

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

**LLM -
Attention
Heads**

LLM -
MLPs

LLM - Un-
embedding

Wrap-up

The Attention Side

Queries and Keys:

- Each token generates a query vector and a key vector.
- For "Amy saw Will," let's assume:
    - "Amy" generates a query looking for a verb, and a key indicating it's a noun.
    - "Saw" queries for subjects and objects, and keys as a verb.
    - "Will" queries for verbs that could precede it, and keys as a noun or modal.

The Attention Side
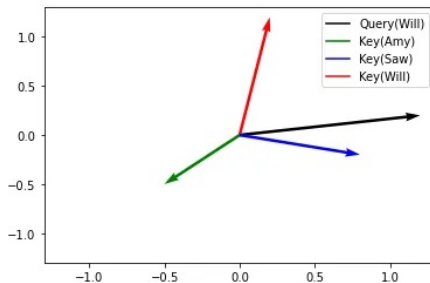
Calculating Attention Scores:

- The model calculates the dot product between queries and keys to produce raw attention scores, indicating the relevance of other tokens.

- For example, "Will's" query might have a higher dot product with "saw's" key, indicating that "Will" should pay more attention to "saw."

Normalization via Softmax:

- The raw scores are then normalized using the softmax function to convert them into a probability distribution.

- This ensures that the attention scores across all tokens sum to one.

- Besides, the model should only allow a token to pay attention to itself and previous tokens. When making predictions about what the next token is, it shouldn't be able to see the future first. So, 'Amy' shouldn't be allowed to attend to 'saw' or to 'Will', and 'saw' shouldn't be allowed to attend to 'Will'.

	Amy	saw	Will
Amy	1	0	0
saw	.51	.49	0
Will	.11	.54	.35

The Value Side

The value side of an attention head dictates the content that is shared between tokens based on the attention scores. This is where the actual 'information transfer' occurs, using the weighted context to update each token's representation.

Week 9

Boris
Babic,
HKU

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

LLM -
Attention
Heads

LLM -
MLPs

LLM - Un-
embedding

Wrap-up

The Value Side

Values:

- Each token also produces a value vector, which holds the information that will be shared with other tokens.

- For instance, the value vector for "saw" might encode information about action and tense.

- Remember the initial embeddings we generate for each token? As with queries and keys, **the attention head comes with its own instructions for how to turn the original embedding into values**.

	Future	Tool	Fame	Male
Amy	0	−.5	.1	−1
saw	−1	.8	.2	0
Will	1	.2	.1	1

	Value$_1$	Value$_2$
Amy	0	−.5
saw	−1	.8
Will	1	.2

The output for the Attention Mechanism

Combining Values with Attention Scores:

- Now, we get two new vectors for each token in addition to the initial embedding vector we generated.
    - This attention scores vector which indicates how much(weight) each token should consider other tokens.
    - The value scores vector which indicates what information should be taken from the other tokens.
- The model will now use the attention scores to figure out how much of the value to move over.
- A token's new representation is a sum of all value vectors, weighted by its attention scores towards other tokens.

The output for the Attention Mechanism

From our attention pattern, we have Will giving 11% of its attention to Amy, 54% of its attention to saw, and 35% to itself. So, it will take 11% of Amy's value, 54% of saw's value, and 35% of Will's value and add that together.

	Amy	saw	Will		Value$_1$	Value$_2$
Amy	1	0	0	Amy	0	$-.5$
saw	.51	.49	0	saw	-1	.8
Will	.11	.54	.35	Will	1	.2

Now we get a new Result vector:

	Result$_1$	Result$_2$
Amy	0	$-.5$
saw	$-.49$	.14
Will	$-.19$	.45

Updating the Embedding vector

Now, we want the output of the attention head to change our original embedding. The model will figure out itself how to add the Result vector to the original embedding vector to get a new embedding for each token.

The updated embeddings will be something like

	Future	Tool	Fame	Male
Amy	0	-1	.1	-1
saw	-1.49	.94	.2	0
Will	.81	.65	.1	1

which has the same structure as the initial embedding vector but now captures the inherent relationships among tokens.

MLPs

After attention heads have updated the embeddings by integrating context, the next step is to pass these vectors through an Multi-Layer Perceptron (MLP). The purpose here is to introduce non-linearity into the process, which is crucial for capturing the complex patterns in language that linear operations cannot.

Boris
Babic,
HKU

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

LLM -
Attention
Heads

**LLM -
MLPs**

LLM - Un-
embedding

Wrap-up

We do the MLP processing in parallel. Each vector (corresponding to a token in the original input) goes through this network on its own. No information is moved from token to token.

Don't worry too much about its inherent mechanism. As of now, this portion is the least understood, in that we have less of an understanding of what it accomplishes in a layer.

MLPs

But why do we have MLPs in addition to attention heads?

The basic reason is that what we get from attention heads are nearly linear. So we're going to want a bit more non-linearity somehow in our model to capture and represent more intricate relationships.

Unembedding: Making Predictions

Finally, we arrive at unembedding, where the model uses the processed vectors to make predictions about the next tokens.

The unembedding takes the highly refined vectors from the MLPs and converts them back into a format suitable for making predictions, which is typically a probability distribution over the next possible tokens.

Unembedding: Making Predictions

Language
Model

Recaps on
NLP

Large
Language
Models

LLM -
Embedding

LLM -
Attention
Heads

LLM -
MLPs

LLM - Un-
embedding

Wrap-up

- As usual, the model has its set of instructions for how to transform the embeddings refined from the MLPs into a final prediction matrix.

- With the final embeddings for "Amy," "saw," and "Will," we apply a transformation to predict the probability of each token in the vocabulary following the given string.

- For instance, based on the vector for "Amy," the model assigns a 20% probability that "Will" is the next word. And after the model sees the vectors for the first three words "Amy saw Will", it predicts that the next word should most likely to be 'Arnett' with a probability of 83%

	Amy	Arnett	Be	Saw	Will	Yesterday
Amy	.18	.07	.18	.18	.20	.18
saw	.29	.03	.03	.11	.03	.51
Will	.00	.83	.06	.03	.07	.01

That's the basic idea of attention heads. You embed the string token-by-token. Then you move information around and change the embedding. Then you do a bit of computation on each token's embedding (without moving information around) with an MLP. Then you keep doing the attention/MLP thing for several iterations to fine-tune the embeddings. Then you make predictions with an unembedding.

NLP v.s. LLM

- Traditional NLP systems often rely on handcrafted features and rule-based methods, which can limit their ability to generalize to new, unseen data or tasks without extensive reprogramming or retraining.

- LLMs generalize better to a variety of tasks without task-specific data.

- LLMs can understand and generate language with a degree of subtlety and complexity that is often difficult for more conventional NLP systems.

- However, LLMs require significant computational resources to train and operate. They may also perpetuate biases present in the training data and can sometimes generate plausible-sounding but incorrect or nonsensical information.

Societal Impacts

While LLMs, like GPT-3 and GPT-4, are more powerful and can perform complex tasks, several concerns arise:

- Job loss or realignment, expansion of disparities
  (Solutions? Forward-looking projections and policy)

- Increase in targeted phishing, fraud, or manipulation
  (Solutions? Stricter verification of information sources and detection of machine-generated text)

- Increased use of algorithmic decisions exacerbating disparities
  (Solutions? Self- and imposed regulation)

Forbes: The Dark Side of ChatGPT

"Promoting a culture of ethics as it relates to AI could be an effective strategy to address bias within the AI systems that are used in the workplace. ... It is no surprise that an AI chat bot like ChatGPT can generate biased responses. But AI technology only mirrors what has been programmed and what it has been trained on. We may be far from the point where we can truly rely on the responses generated from any AI system. While the possibilities of AI systems are limitless, they must be used with a grain of salt and an understanding of their potential limitations."

Source:https://www.forbes.com/sites/janicegassam/2023/01/28/the-dark-side-of-chatgpt/?sh=533828064799