

# CMPE492 - Week 1

Zeynep Buse Aydın - Asım Dağ

# LLM

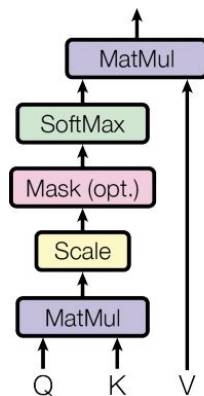
- a deep learning algorithm, neural network
  - can perform a variety of NLP tasks
  - uses **transformer** architecture and is trained using **massive** datasets
- 
- more complex tasks (text generation and recognition)
  - generalizing better across different language tasks
  - better at understanding text because the other models process sequentially, word by word

# Transformers

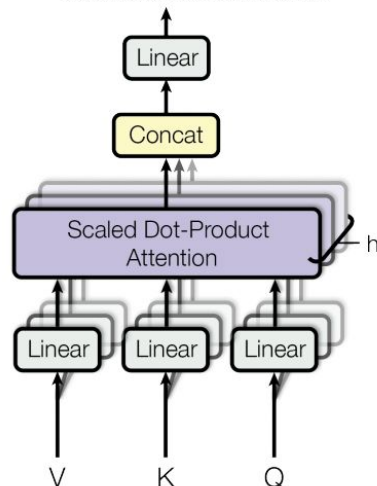
## Self Attention

- Calculates a weighted sum of all the other words' representations, allowing the model to weigh the importance of each word in a sentence relative to all others.
- Understand context over long distances in text.
- Can handle the entire input sequence simultaneously. Does the job faster and can handle more complicated connections between words in the input sequence.
- Lets the model selectively focus on different parts of the input sequence instead of treating everything the same way.

Scaled Dot-Product Attention



Multi-Head Attention



Attention Is All You Need, <https://arxiv.org/pdf/1706.03762>

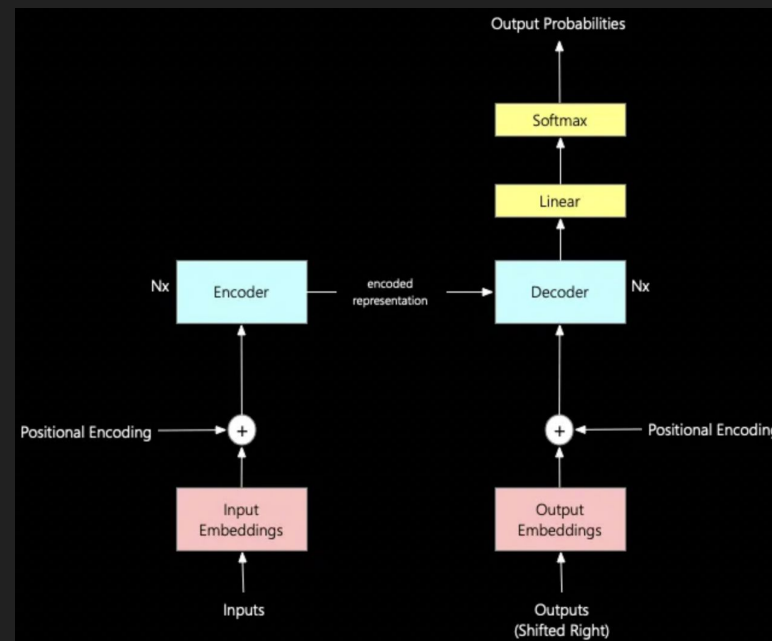
# Transformers - continued

## Encoder - Decoder structure

**Positional Encoding:** keep track of word order: embeddings + positional values

## Multi-Headed Attention

**Feed-Forward Layers:** Fully connected feed-forward neural networks transform the data into more abstract representations. Introduce non-linearity into the model through activation functions



<https://pradeepmenon.medium.com/introduction-to-large-language-models-and-the-transformer-architecture-534408ed7e61>

# Pre-Training vs Fine-Tuning

## Pre-Training

- Pre-training usually would mean take the original model, initialise the weights randomly, and train the model from absolute scratch on some large corpora.
- Expensive
- Massive dataset
- Unsupervised Learning
- Pre-training allows the model to leverage transfer learning, where knowledge gained from one domain (general language) can be applied to various tasks without having to start training from scratch each time.

## Fine-Tuning

- Fine-tuning employs **labeled data** to fine-tune the model's parameters, tailoring it to the **specific nuances of a task**. This **specialization** significantly enhances the model's effectiveness in that particular task compared to a general-purpose pre-trained model.
- Task-Specific small datasets
- Supervised Learning with task specific loss functions

# Tokenization

- Breaking down text into smaller, manageable units called tokens, such as words, subwords, or characters, which LLMs (Large Language Models) can then process and understand.
- Word-level tokenization, subword-level tokenization (like Byte Pair Encoding or WordPiece), and character-level tokenization.
- **Efficient Processing:** computationally less expensive
- **Improved Understanding:** especially subword tokenization allows the LLM to capture the finer details of language
- **Adaptability:** Tokenization, especially subword tokenization, allows LLMs to tackle unfamiliar terms by relating them to known elements.
- **Enhanced Prediction Capabilities:** By considering smaller units, the LLM can potentially build more accurate statistical models of language patterns.
- **More granular** methods induce more token data, meaning it will need more memory, more processing power, and more time downstream
- **Coarser** methods consume less computational resources but would risk the model losing the semantics and expressiveness in meaning.

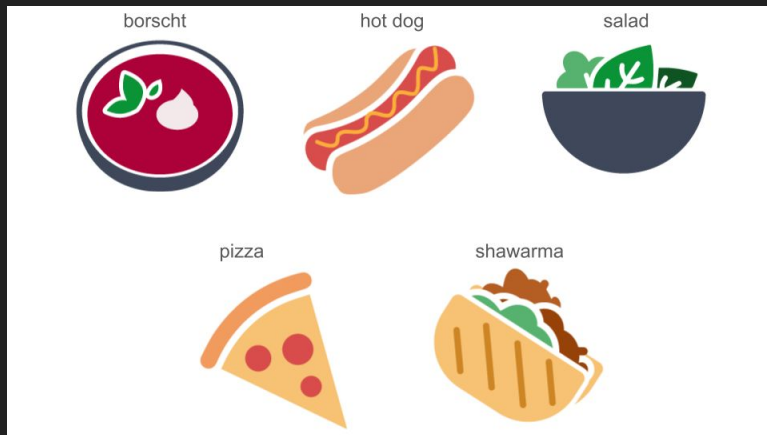
# Tokenization – example

## Byte Pair Encoding(BPE)

- The core of GPT models. BPE was originally a data compression technique from 1994, but later applied to tokenization by Senrich in 2016.
- Iteratively searches for frequent pairs of characters and merges them into a new symbol. The symbols are continuously added to the vocabulary until a sufficient vocabulary size is constructed.
- Proficient at compressing text, meaning larger amounts of text can be represented by shorter sequences of tokens.
- Effectively helps the language model understand random, unfamiliar words, mitigating the issues associated with 'Out-of-Vocabulary(OOV)' words. This is especially useful in specialized domain texts.

# Embeddings



















- Dense, continuous vector representations of words or tokens
- **Purpose:** Capturing semantic relationships (similar words have similar embeddings)
- Math behind embeddings is basically converting from one-hot vector to a dense vector in lower dimensions.
- Lets give an example: here are foods dataset

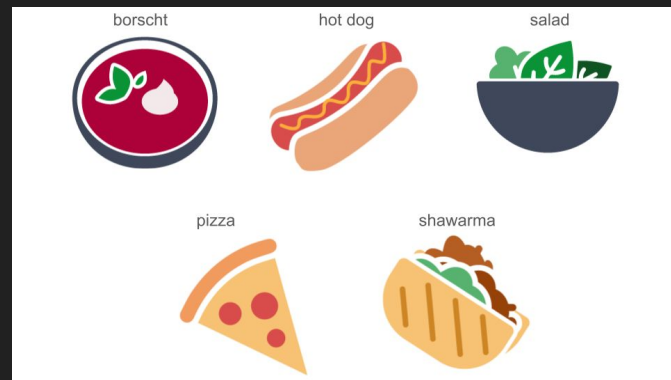




# Embeddings - Example

- This is an example **one-hot encoded** representation of our foods

borscht						
[	1,	0,	0,	0,	...,	0]
						
hot dog						
[	0,	1,	0,	0,	...,	0]
						
shawarma						
[	0,	0,	0,	0,	...,	1]
						

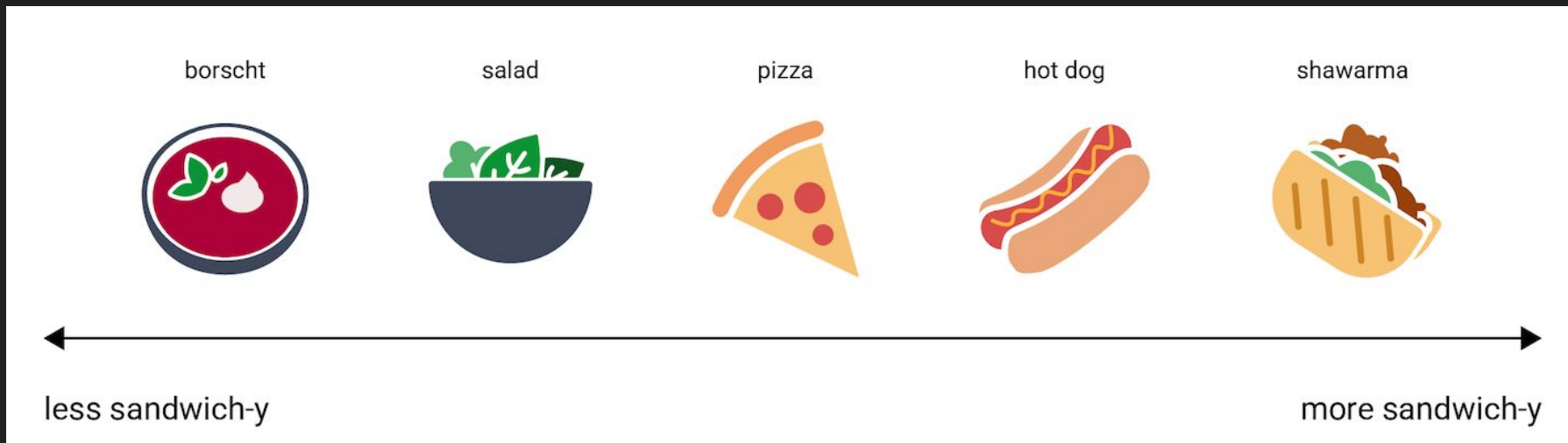
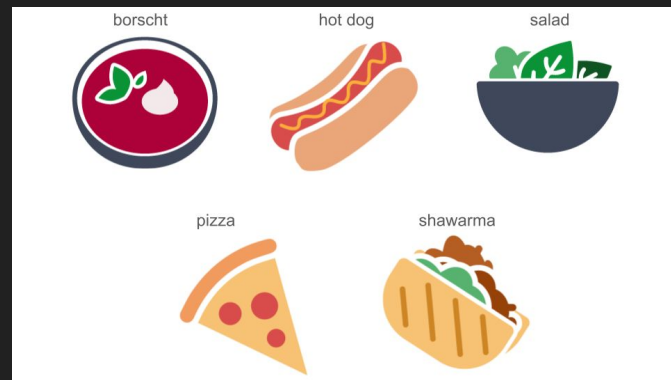


Why we need to **lower** dimensions?

- Number of weights (Large input vectors mean a huge number of weights)
  - The more weights -> the more data we need
  - The more data -> time/memory inefficiency
- Lack of meaningful relations between data

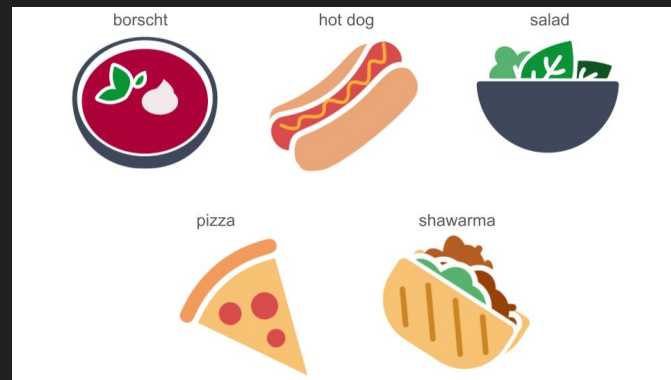
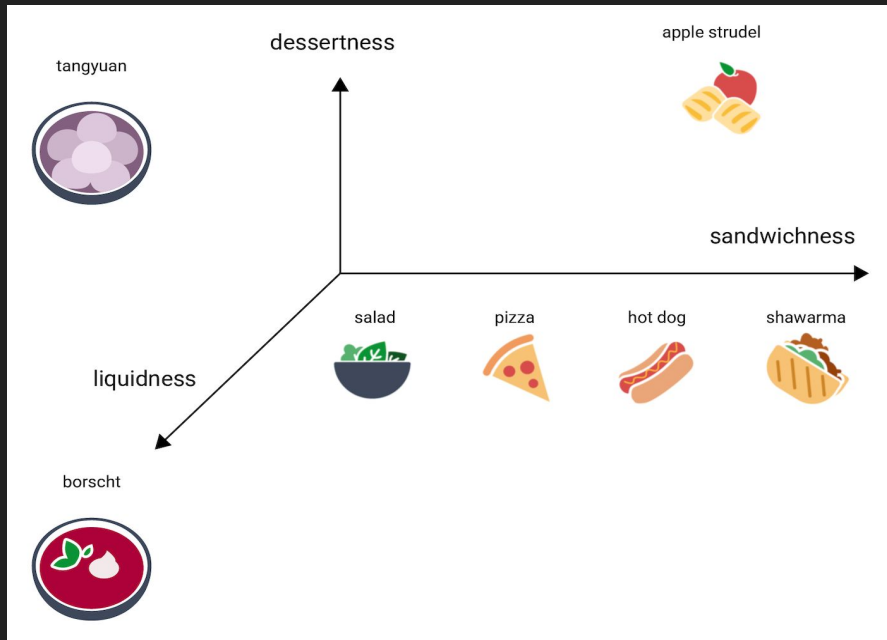
# Embeddings - Example (continued)

- Lets consider **one-dimensional representation** of foods on the scale of “least like a sandwich”



# Embeddings - Example (continued)

- By adding more dimensions we can group each food semantically and keep dissimilar items far apart.

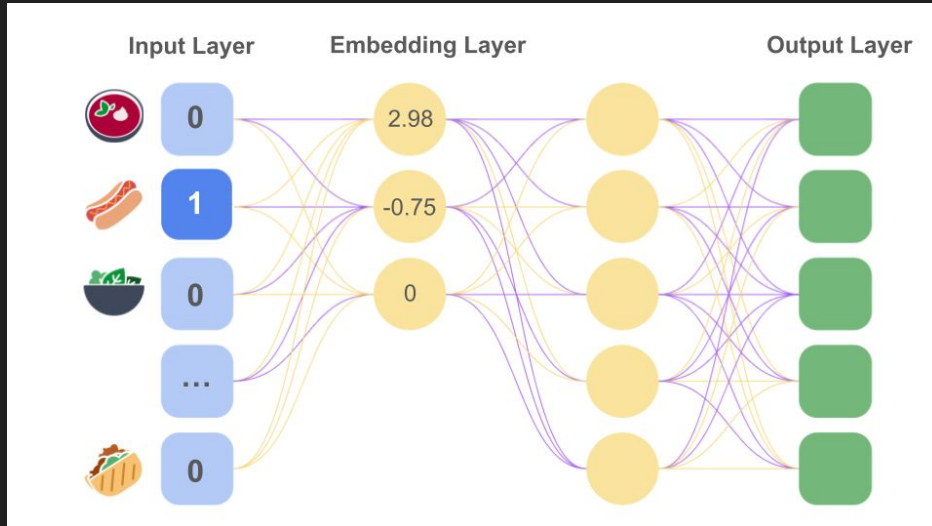


Similarities of each word, each food in our case, can be measured using cosine similarities function:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}},$$

# Embeddings in NN

- When creating neural networks, embeddings can be hidden layers between input layer and output layer.
- During training, the model will learn the optimal weights for the nodes in hidden layers, ie. embedding layers



In our example, let's say at embedding layer, three most relevant dimensions for foods are *sandwichness*, *desertness* and *liquidness*.

A one-hot encoding of hot dog provided as input, translates it into three dimensional embedding vector  $[2.98, -0.75, 0]$

# Evaluation of LLMs

## 1. Perplexity:

- Means how “surprised” or “confused” the model is when it predicts the next word in a sequence.
- Can be calculated using this formula:

$$\text{PPL} = \exp \left( -\frac{1}{T} \sum_{t=1}^T \log P(w_t | w_1, w_2, \dots, w_{t-1}) \right)$$

- T is the total number of tokens in the sequence
- $P(w_t | w_1, w_2, \dots, w_{t-1})$  is the probability assigned by the model to token  $w_t$  given the preceding tokens.

If the model assigns high probability to the correct word  $w_t$ ,  $\exp$  will be closer to the 0, so lower ppl means higher probability, hence correct guess.

# Evaluation of LLMs

## 2. ROUGE(Recall-Oriented Understudy for Gisting Evaluation)

- Measures the recall of n-grams from reference text in the generated summary.
- typically used for evaluating summarization tasks.

## 3. BLEU(Bilingual Evaluation Understudy)

- Measures the overlap of n-grams between the generated text and human reference text.
- Typically used for tasks like translation.

# Evaluation of LLMs

## 4. Accuracy

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

For classification tasks (like sentiment analysis or text classification), accuracy is used to measure the percentage of correct predictions.

# Evaluation of LLMs

## 5. Human Evaluation

- Perplexity, ROUGE, BLEU, Accuracy are objective metrics, some tasks require human judgement.
- Especially critical for:
  - Fluency: Does the generated text sound natural?
  - Coherence: Does the generated text make sense and stay on topic?
  - Relevance: Does the generated text directly address the input query?



# Real World Applications of LLMs

- Virtual Assistants (Customer Support)
- Chatbots (Alexa, Siri, ChatGPT)
- Content Generation (News, magazines)
- Code Generation and Debugging (Cursor, Github Copilot)
- Healthcare (Neyim var)
- Translations
- Summarizations of articles
- Author assistants

App Demos

<https://www.elastic.co/what-is/large-language-models>

<https://medium.com/@lktstdvd/the-difference-between-large-language-models-llms-and-traditional-machine-learning-models-c338af4b01b3>

<https://rpradeepmenon.medium.com/introduction-to-large-language-models-and-the-transformer-architecture-534408ed7e61>

<https://medium.com/@eordaxd/fine-tuning-vs-pre-training-651d05186faf>

<https://medium.com/@younesh.kc/exploring-tokenization-the-backbone-of-modern-language-models-87b433a7c41e>