# EEP 596: LLMs: From Transformers to GPT || Lecture 7

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 29, 2025

# Deep Learning References

### Deep Learning

Great reference for the theory and fundamentals of deep learning: Book by Goodfellow and Bengio et al Bengio et al
Deep Learning History

### Embeddings

SBERT and its usefulness SBert Details

# Last lecture

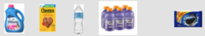
- Recap of Embeddings and Cosine Similarity
- Glove Embeddings
- Sentence Embeddings with Glove and Sentence Transformer
- Product2Vec

# Today's Lecture

- Product2Vec Recap
- Embedding Theory
- Sentence Transformers
- Vector Databses

# Product2Vec

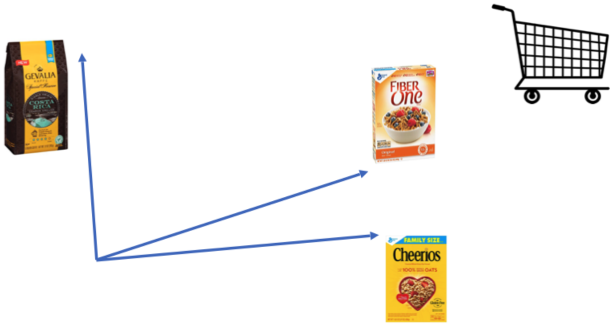

Represent products in product space with a large matrix of embedding coordinate vectors ***"L"***



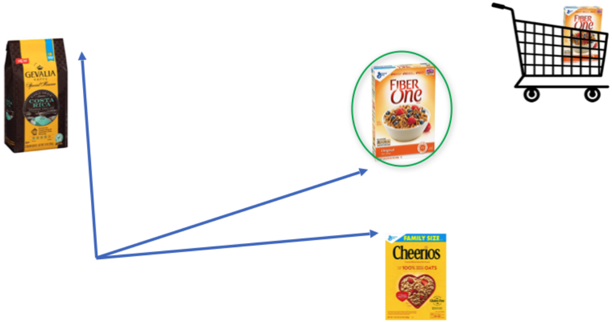
$$L = \begin{pmatrix} 1.5 & 1.9 & 1.8 & 1.4 & \cdots & 0.4 \\ 0.6 & 0.1 & 1.0 & 1.6 & \cdots & 1.9 \\ 0.6 & 1.6 & 1.6 & 1.6 & \cdots & 1.8 \\ 0.6 & 1.0 & 0.1 & 1.6 & \cdots & 0.6 \\ 0.8 & 1.4 & 1.9 & 0.8 & \cdots & 0.7 \end{pmatrix}$$

We obtain these embedding vectors from the Product2Vec service [London et al, 2017]

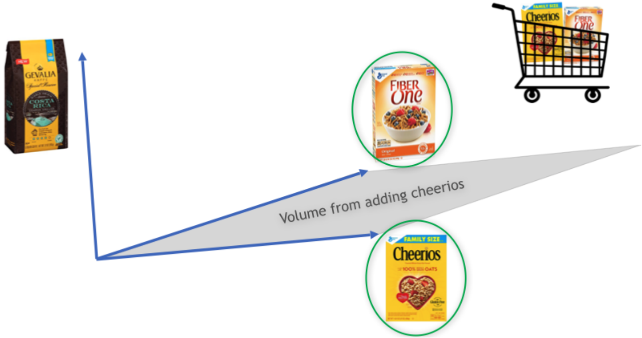
# Product2Vec application



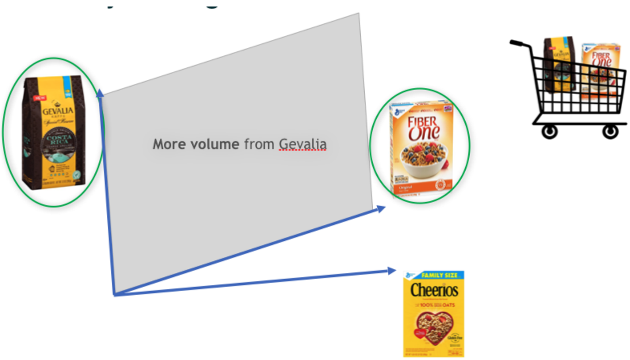
# Product2Vec application



# Product2Vec application



# Product2Vec application

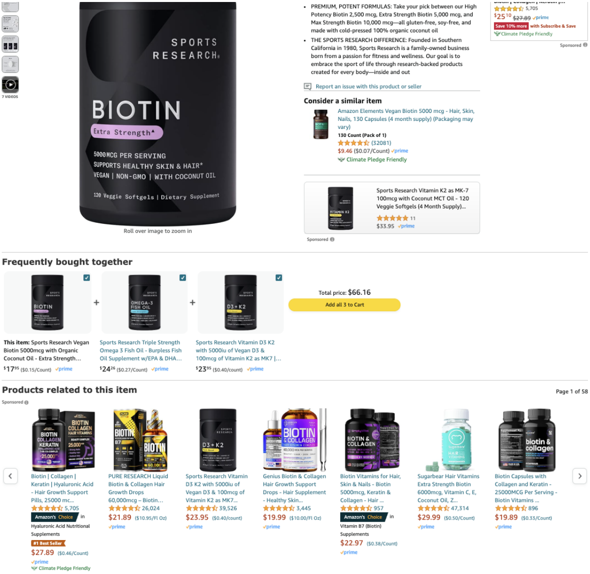


# ICE #1

### View Similarity or Purchase Similarity

Consider a company that sells products online. As we know, embedding representations for words or products in this case are learned from data. The question is which data to use? These are referred to as signals sometimes. So the question is, does **view similarity** of products represent a better signal for learning embeddings or **purchase similarity**? Remember: Good embeddings embed similar products close to each other dis-similar products away from each other.

# ICE #1: View or Purchase Similarity?



# Generating Sentence Embeddings from Glove

**Averaging embeddings of words:** If we have a word embedding, how do we generate the sentence embedding?

# Generating Sentence Embeddings from Glove

**Averaging embeddings of words:** If we have a word embedding, how do we generate the sentence embedding?

**Simple Solution: Just average the word embeddings**

# How do we improve Sentence Embeddings?

## Sentence Embeddings

As you are probably observing in your Mini-Project 1 assignment - Averaging word embeddings doesn't "perform" as well. So we need sentence embeddings that do better than just averaging word embeddings - Perhaps, capture the sequence of information flow in a sentence.

# How do we improve Sentence Embeddings?

## Sentence Embeddings

As you are probably observing in your Mini-Project 1 assignment - Averaging word embeddings doesn't "perform" as well. So we need sentence embeddings that do better than just averaging word embeddings - Perhaps, capture the sequence of information flow in a sentence.

## Example 1

Sentence 1: "Me loves my friend"
Sentence 2: "My friend loves me"
Should they have the exact same sentence embeddings?

# How do we improve Sentence Embeddings?

### Sentence Embeddings

As you are probably observing in your Mini-Project 1 assignment - Averaging word embeddings doesn't "perform" as well. So we need sentence embeddings that do better than just averaging word embeddings - Perhaps, capture the sequence of information flow in a sentence.

### Example 1

Sentence 1: "Me loves my friend"
Sentence 2: "My friend loves me"
Should they have the exact same sentence embeddings?

### Example 2

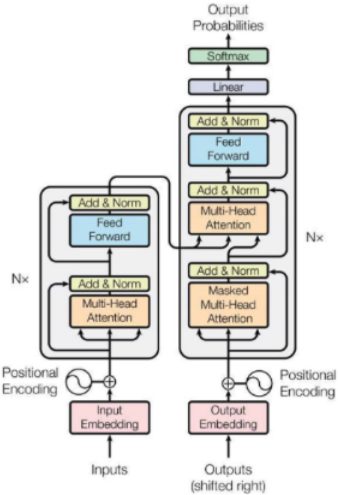
Sentence 1: "I like chocolate milk"
Sentence 2: "I like milk chocolate"
Should they have the same sentence embeddings?

# Next Topic: Transformers, BERT and connections to Embeddings

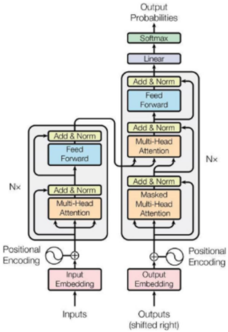
## Capturing Sequence of information

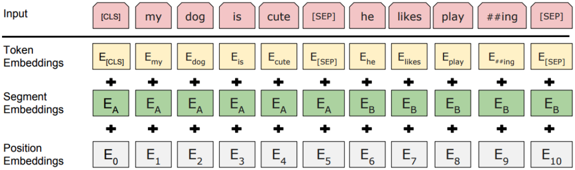
As we discussed in the history of Deep Learning - RNNs and LSTMs are DL archs that are able to capture sequence information in a sentence to some extent (the **chocolate milk** vs **milk chocolate** example). On the other hand, they weren't robust to larger context or multiple sentences and could only operate with smaller sentence lengths. This is where the advent of Transformers was a breakthrough for ML/DL and AI in general - They could do much better in capturing context, sequential information, supported multiple sentences and paragraphs, etc.

# Transformers - Encoder and Decoder Architecture

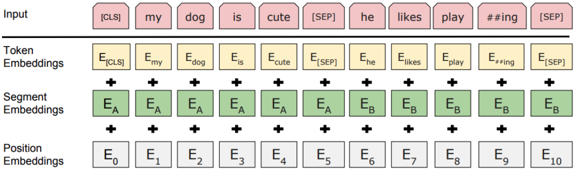


# Encoder and Encoder Embeddings



# Understanding Encoder/BERT at high-level



# BERT - Bi-directional Encoders from Transformers

| System | MNLI-(m/mm) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.8 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 87.4 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.1 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **92.7** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **82.1** |

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Encoder:** The **architecture component** of the transformer that transforms inputs through a series of Neural layers into a vector (embedding). This vector can then be useful for downstream tasks: Emotion detection, Classification, etc

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Encoder:** The **architecture component** of the transformer that transforms inputs through a series of Neural layers into a vector (embedding). This vector can then be useful for downstream tasks: Emotion detection, Classification, etc
2. **Decoder:** The **architecture component** of the transformer that transforms inputs one at a time to generate words in sequence. Example: You ask a question of ChatGPT and it starts generating words, one at a time - Just like it were typing. That's decoder in action.

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Encoder:** The **architecture component** of the transformer that transforms inputs through a series of Neural layers into a vector (embedding). This vector can then be useful for downstream tasks: Emotion detection, Classification, etc
2. **Decoder:** The **architecture component** of the transformer that transforms inputs one at a time to generate words in sequence. Example: You ask a question of ChatGPT and it starts generating words, one at a time - Just like it were typing. That's decoder in action.
3. **Input Embedding:** How an input gets embedded as a vector. What would be the starting point to embed "chocolate milk" as a vector?

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Encoder:** The **architecture component** of the transformer that transforms inputs through a series of Neural layers into a vector (embedding). This vector can then be useful for downstream tasks: Emotion detection, Classification, etc
2. **Decoder:** The **architecture component** of the transformer that transforms inputs one at a time to generate words in sequence. Example: You ask a question of ChatGPT and it starts generating words, one at a time - Just like it were typing. That's decoder in action.
3. **Input Embedding:** How an input gets embedded as a vector. What would be the starting point to embed "chocolate milk" as a vector?
4. **Token Embedding:** This refers to the individual token embeddings or word embedddings (or sub-word embeddings)

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Segment Embedding:** This refers to a generic embedding that says this was segment 1 or segment 2 of the input

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Segment Embedding:** This refers to a generic embedding that says this was segment 1 or segment 2 of the input
2. **Position Embedding:** This adds in the position information into the embedding. Did "chocolate" come in at the beginning of the sentence or middle or the end?

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Segment Embedding:** This refers to a generic embedding that says this was segment 1 or segment 2 of the input
2. **Position Embedding:** This adds in the position information into the embedding. Did "chocolate" come in at the beginning of the sentence or middle or the end?
3. **BERT Embedding:** This is the embedding or the vector used after having gone through the Encoder Architecture

# Parsing the Embeddings and Encoder/Decoder Terminology

1. **Segment Embedding:** This refers to a generic embedding that says this was segment 1 or segment 2 of the input
2. **Position Embedding:** This adds in the position information into the embedding. Did "chocolate" come in at the beginning of the sentence or middle or the end?
3. **BERT Embedding:** This is the embedding or the vector used after having gone through the Encoder Architecture
4. **Sentence BERT (sBERT) Embedding:** This is the embedding that you are using in Mini-Project 1, an **encoding** into a vector that's optimized for sentence similarity! (More on this in a bit)

# Linear Transformations and Projections

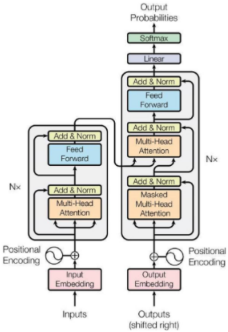
# Path of a sentence through transformer layers

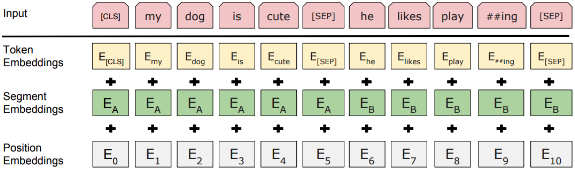
# Llama3 High-level Numbers

## Llama 3 70B Model

- Parameters: Approximately 70 billion.
- Vocabulary Size: 128,000 tokens.
- Total tokens seen: 15 trillion
- Context Window: Supports sequences up to 8,192 tokens.
- Architecture:
- Layers: 80 transformer blocks.
- Attention Heads: 64 heads per layer.
- Feature Dimension: 8,192.
- Hidden Dimension: 10,649.

# Encoder and Encoder Embeddings



# In-class Coding exercise - Carries Bonus points for grade

### Compute Sentence Embedding (15 mins)

**Instructions:** Write a python module to compute sentence embeddings given words and their embeddings.
Encapsulate your code in a class structure. With one method for each of the sentence embedding computations asked below. Each method takes in as input a sentence and outputs a sentence embedding. Also have class variables to store vocab embeddings, filler words and attention weights (part 3).
**Submit:** Your code on canvas link that will be opened up shortly for bonus points that can be used towards your grade.
**Vocabulary:** Assume the vocabulary consists of 5 words:
$I, lot, love, chocolate, milk$.
**Sentence:** Consider the following sentence: `I love chcolate milk as well!` Assume the embedding(ith word) $= [i-1, i+1]$ where $i$ is 0-indexed.

# In-class Coding exercise - Carries Bonus points for grade

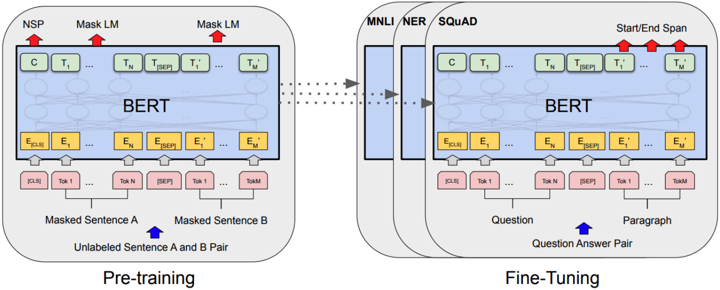
## Compute Sentence Embedding (15 mins)

**1. Simple Average Embedding:** Compute the sentence embedding by simple averaging. For any word not in vocabulary, use the embedding as $[-1, -1]$. What's the resulting sentence embedding with this approach?

**2. Skip filler words embedding:** Let filler words be $[I, and, as, for, it, or, maybe]$. If you encounter any filler words in a sentence, skip the filler word for the embedding computation. What's the sentence embedding with this approach?

**3. Learned Sentence embedding:** Assume you learned a set of weights that in word embeddings as input and give out sentence embedding as output. This is done by taking a weighted average of the embeddings. For the above sentence, let the weights learned could be from a self-attention layer. Assume the weights for words in the vocabulary are: $[0.5, 1, 0.7, 0.9, 0.3, 0.2, 0.1]$. What would be the resulting embedding?

# BERT - Bi-directional Encoders from Transformers



# BERT pre-training

## Two Tasks

1. **Masked LM Model:** Mask a word in the middle of a sentence and have BERT predict the masked word
2. **Next-sentence prediction:** Predict the next sentence - Use both positive and negative labels. How are these generated?

# BERT pre-training

## Two Tasks

1. **Masked LM Model:** Mask a word in the middle of a sentence and have BERT predict the masked word
2. **Next-sentence prediction:** Predict the next sentence - Use both positive and negative labels. How are these generated?

## ICE: Supervised or Un-supervised?

1. Are the above two tasks supervised or un-supervised?

# BERT pre-training

## Two Tasks

1. **Masked LM Model:** Mask a word in the middle of a sentence and have BERT predict the masked word
2. **Next-sentence prediction:** Predict the next sentence - Use both positive and negative labels. How are these generated?

## ICE: Supervised or Un-supervised?

1. Are the above two tasks supervised or un-supervised?

## Data set!

English Wikipedia and book corpus documents!

# Loss Function for Masked Language Model (MLM)

Loss Function for MLM mimicks which type of classic ML model?

# Loss Function for Masked Language Model (MLM)

Loss Function for MLM mimicks which type of classic ML model?

Cross-Entropy

$L(p, \hat{p}) = -\sum_i \left[p_i \log(\hat{p}_i) + (1 - p_i) \log(1 - \hat{p}_i)\right]$

# Loss Function for Masked Language Model (MLM)

Loss Function for MLM mimicks which type of classic ML model?

Cross-Entropy

$L(p, \hat{p}) = -\sum_i \left[p_i \log(\hat{p}_i) + (1 - p_i) \log(1 - \hat{p}_i)\right]$

ICE: What is the loss function for Binary Classification?

# BERT - Bi-directional Encoders from Transformers

| System | MNLI-(m/mm) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.8 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 87.4 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.1 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **92.7** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **82.1** |

# Sentence BERT a.k.a sBERT

Uses Siamese Twins architecture

# Sentence BERT a.k.a sBERT

Uses Siamese Twins architecture

Advantages of sBERT

More optimized for Sentence Similarity Search.

# SBERT - Siamese BERT architecture

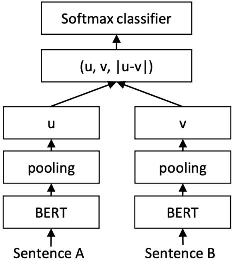



Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

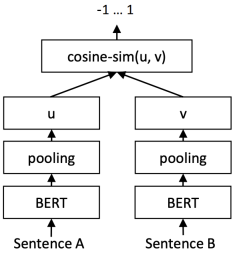



Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

# Loss Function for SBERT

# Breakouts Time #2

### Retrieving Tables with Chat bots — 7 mins

You are building a chat-bot product at your company where queries come in from customers that own data in your company's cloud service. Your chat-bot responds and retrieves the right table or combination of tables (through merge/filter operations) that contains this information or returns back with follow up questions to get more precise information or get back with a "Sorry, I don't have that information" response. How would you go about building a chat-bot like this? What data would you use? What data stores/data bases would be appropriate? What Deep Learning models would you use, would it be supervised or un-supervised learning? What would be your evaluation metric? How would you test if your chat bot is accurate in its responses?