

EEP 596: LLMs: From Transformers to GPT || Lecture 6

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 23, 2024

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](#) [Instacart Search Relevance](#)
[Instacart Auto-Complete](#)

Attention

[Illustration of attention mechanism](#)

Last Lecture

- Product2Vec and X2Vec - Industry use-cases
- Embedding Theory
- Sentence Transformers (BERT)

Today's Lecture

- Sentence Transformers and BERT
 - Loss functions for BERT
 - Multi-Head Attention
 - SBERT
 - (Application of Embeddings to Autocomplete and Search Relevance)
- } Foundations

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"

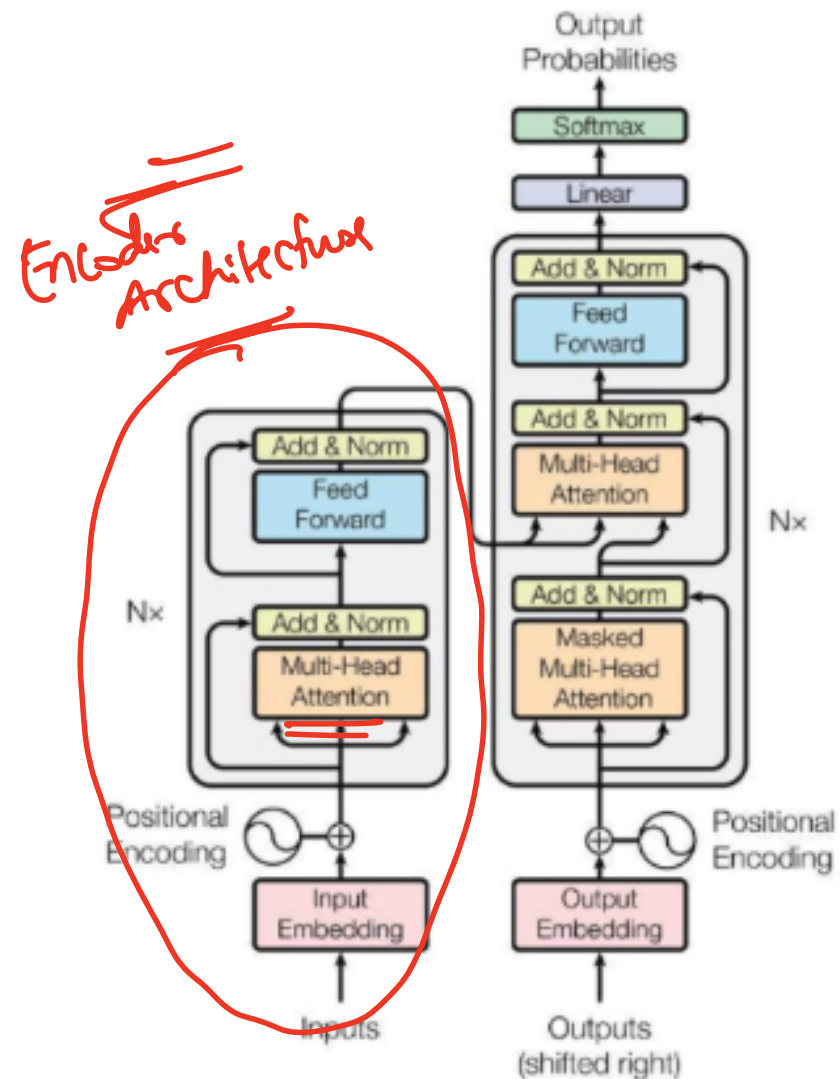
Should they have the same sentence embeddings?

Next Topic: Transformers, BERT and connections to Embeddings

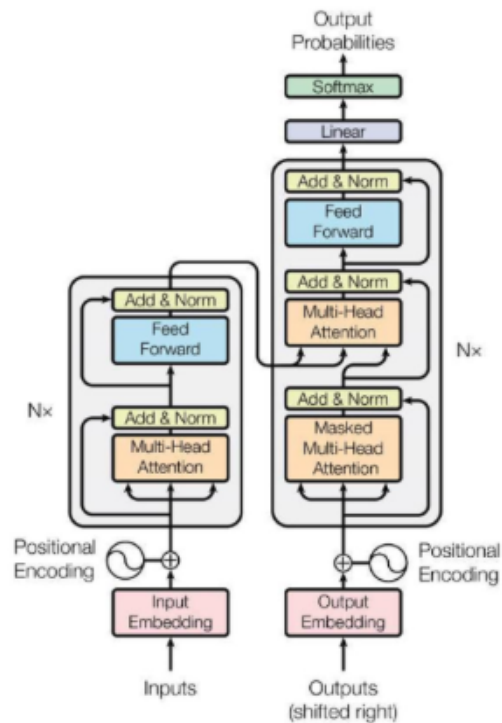
Capturing Sequence of information

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



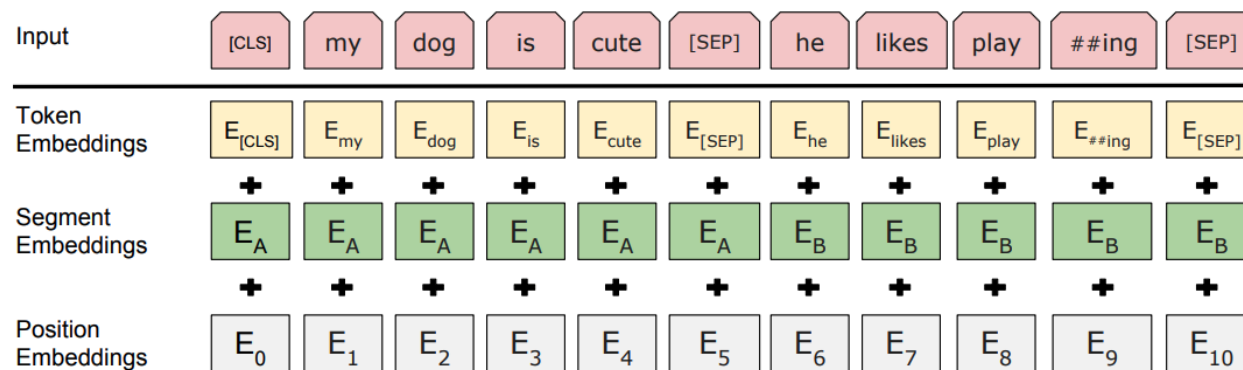
Encoder and Encoder Embeddings



| | | | | | | | | | | | |
|---------------------|-------------|----------|-----------|----------|------------|-------------|----------|-------------|------------|-------------|-------------|
| Input | [CLS] | my | dog | is | cute | [SEP] | he | likes | play | ##ing | [SEP] |
| Token Embeddings | $E_{[CLS]}$ | E_{my} | E_{dog} | E_{is} | E_{cute} | $E_{[SEP]}$ | E_{he} | E_{likes} | E_{play} | $E_{##ing}$ | $E_{[SEP]}$ |
| | + | + | + | + | + | + | + | + | + | + | + |
| Segment Embeddings | E_A | E_A | E_A | E_A | E_A | E_A | E_B | E_B | E_B | E_B | E_B |
| | + | + | + | + | + | + | + | + | + | + | + |
| Position Embeddings | E_0 | E_1 | E_2 | E_3 | E_4 | E_5 | E_6 | E_7 | E_8 | E_9 | E_{10} |

→ Locating a word

Understanding Encoder/BERT at high-level



BERT - Bi-directional Encoders from Transformers

| System | MNLI-(m/mm) 392k | QQP 363k | QNLI 108k | SST-2 67k | CoLA 8.5k | STS-B 5.7k | MRPC 3.5k | RTE 2.5k | Average - |
|-----------------------|---------------------|-------------|--------------|--------------|--------------|---------------|--------------|-------------|--------------|
| 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 |
| BERT _{BASE} | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| BERT _{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

Architecture

- 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

BERT ~ Transformer
no ~ Encoding
Bi-directional

\$BERT = Sentence
Transformer



SBERT = Sentence
Transformer

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?

→ Token Embed + Positional Embed + Segment Embed

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

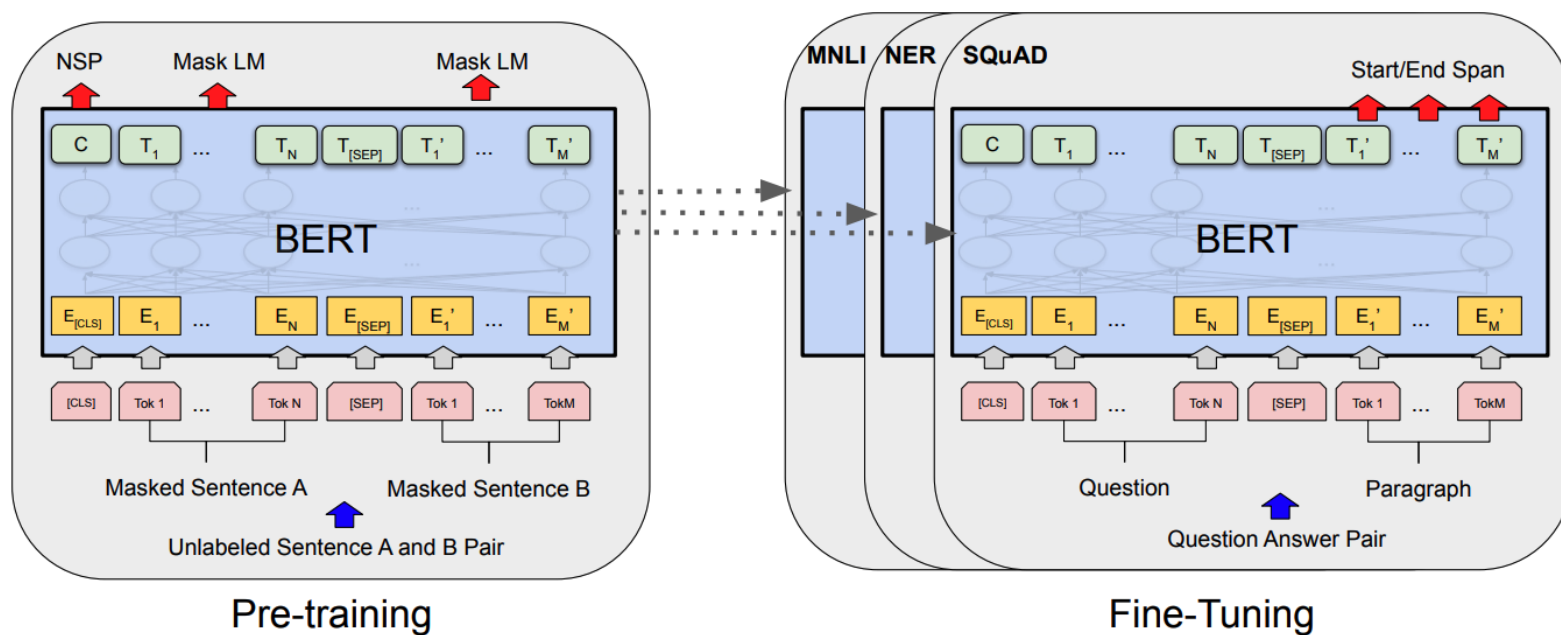
- ① **Segment Embedding:** This refers to a generic embedding that says this was segment 1 or segment 2 of the input
- ② **Position Embedding:** This adds in the position information into the embedding. Did “chocolate” come in at the beginning of the sentence or middle or the end?
- ③ **BERT Embedding:**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

BERT Embedding

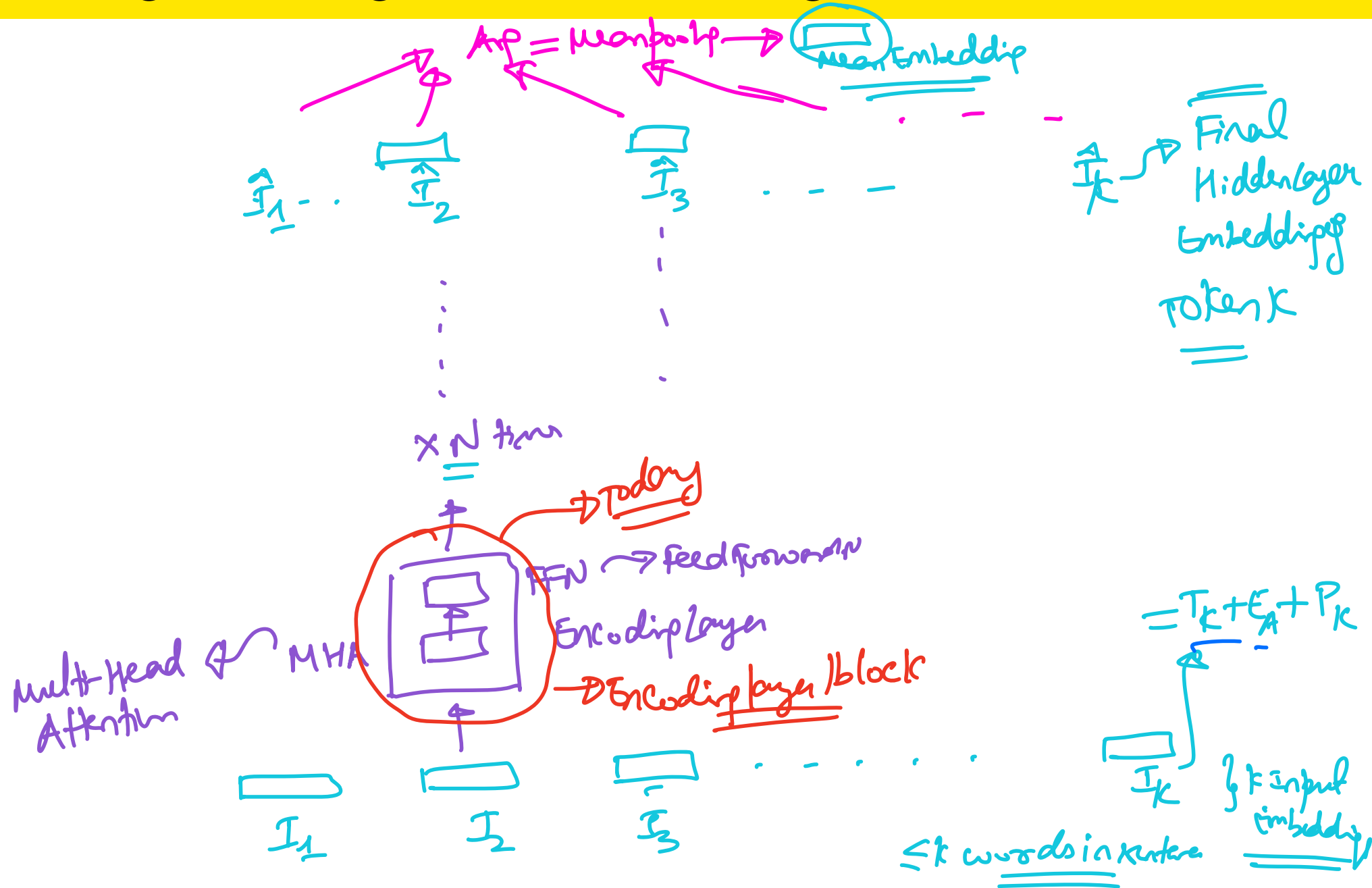
Parsing the Embeddings and Encoder/Decoder Terminology

- 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)

BERT - Bi-directional Encoders from Transformers



Parsing Encoding Transformers: Big Picture

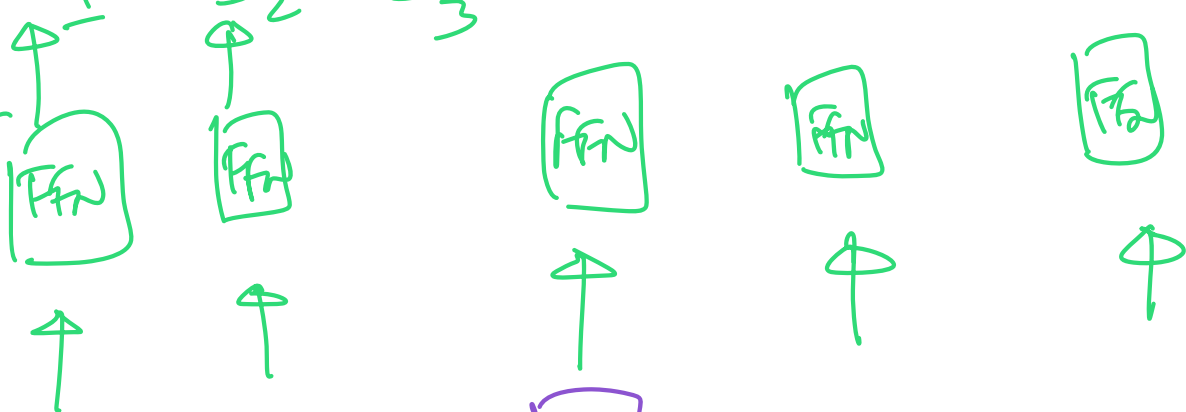


\vec{z}_1^2 \vec{z}_2^2



I_1^2 I_2^2 I_3^2 . . . I_k^2

parallel

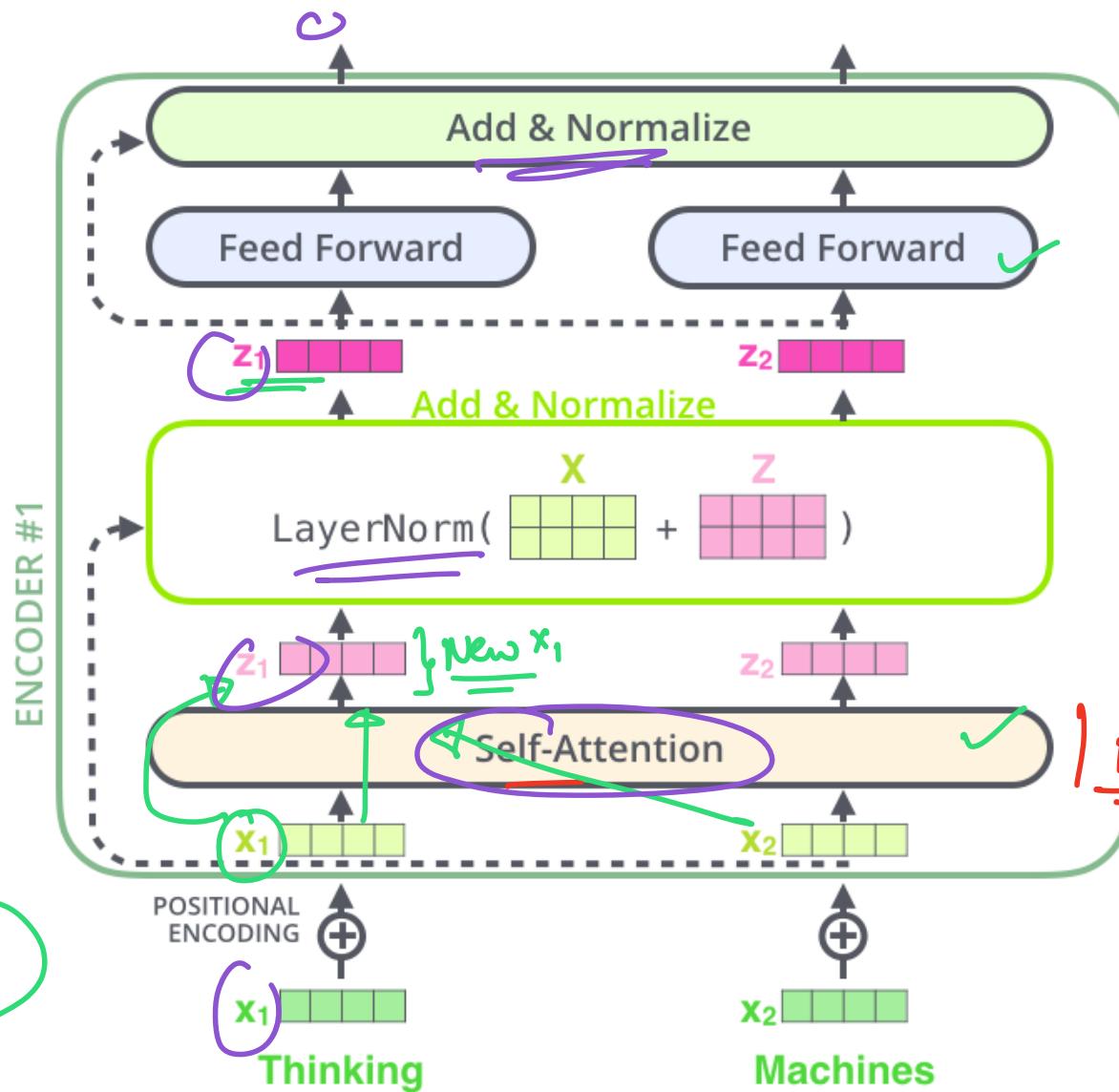


\vec{z}_1 \vec{z}_2 . . . \vec{z}_k



$\} k \text{ tokens}$
 head embeddings

Parsing Encoder: Multi-Head Attention and FFN



Two kinds of attention:-

1) Encoder-Decoder Attention

2) Self Attention

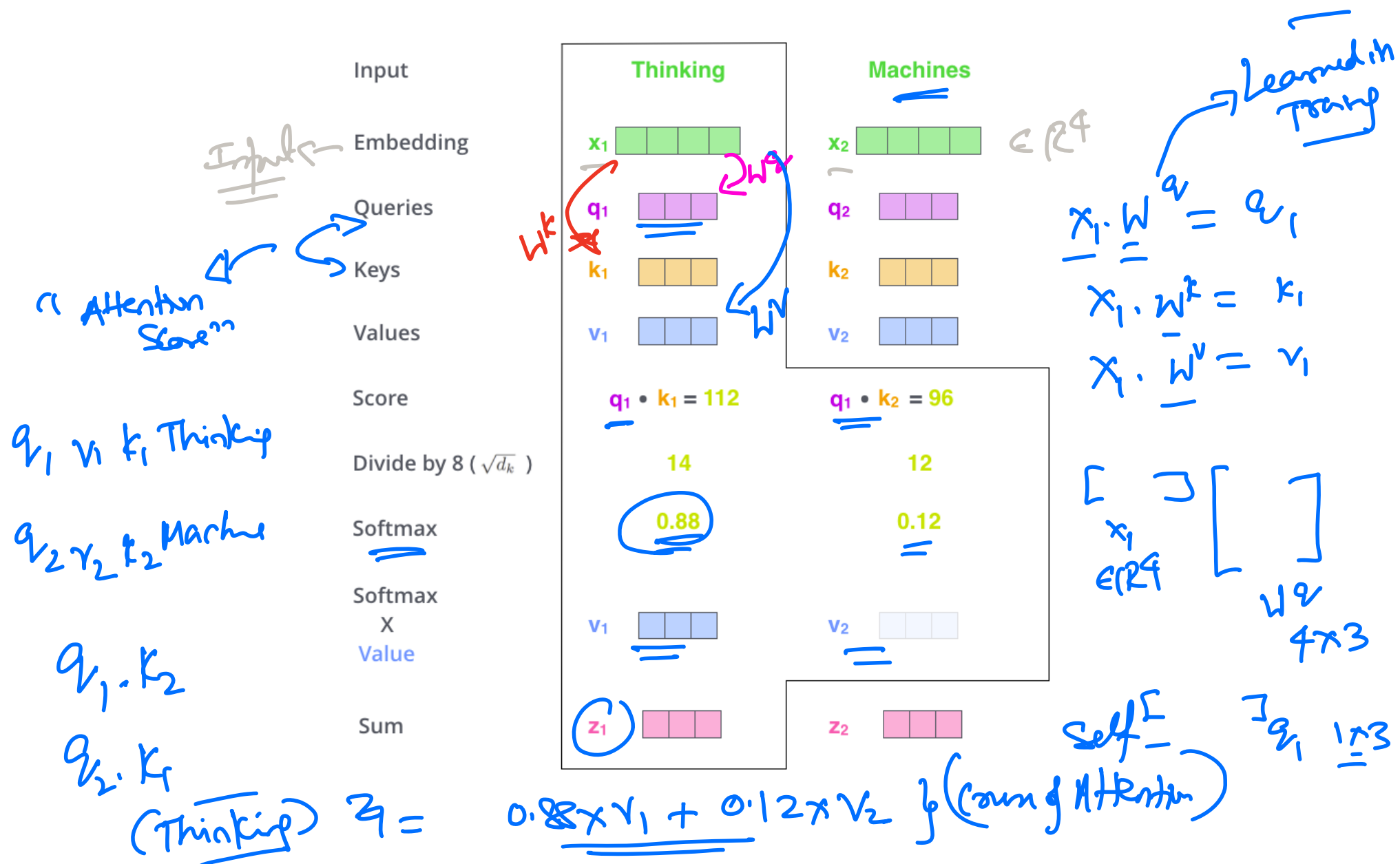
Parallelize

Faster
 \rightarrow Transp

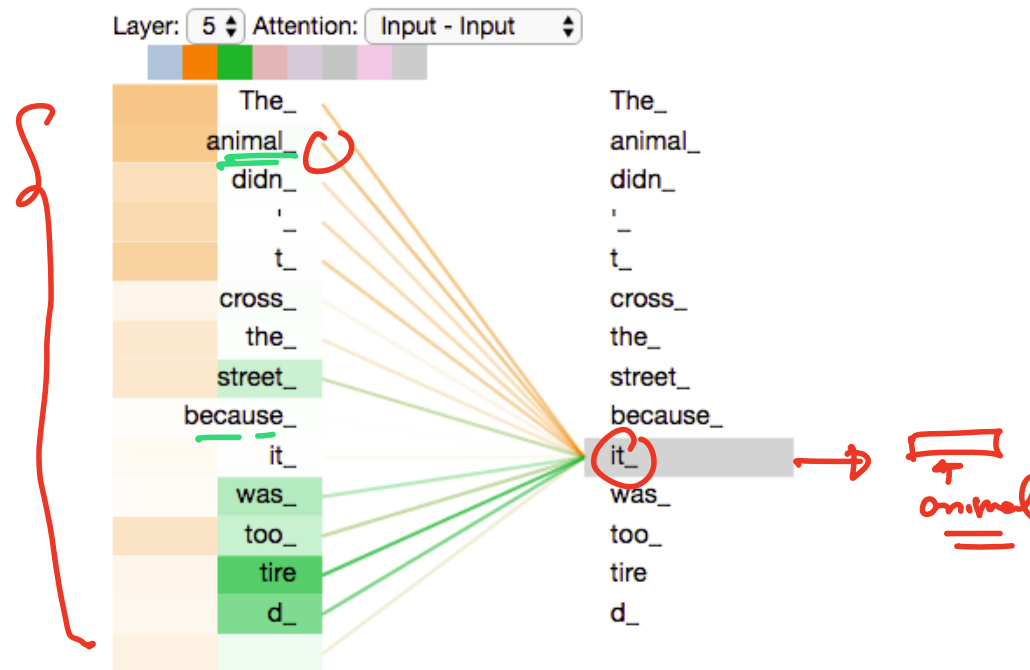
comp. bottle neck
 $O(N^2)$
complex

LoRA

Parsing Encoder: Multi-Head Attention and FFN

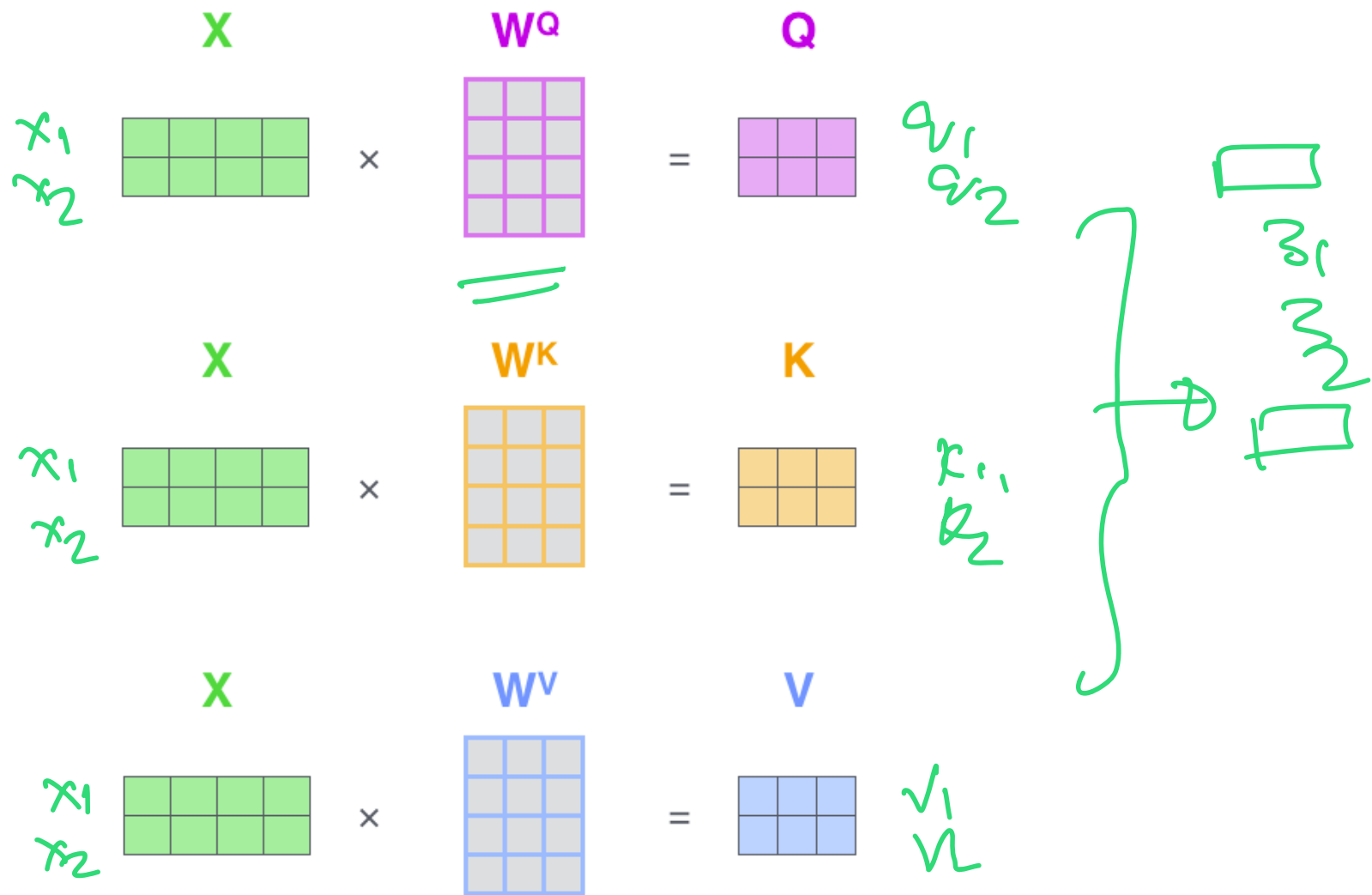


Parsing Encoder: Multi-Head Attention and FFN



As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

Parsing Encoder: Multi-Head Attention and FFN



Every row in the X matrix corresponds to a word in the input sentence. We again see the difference in size of the embedding vector (512, or 4 boxes in the figure), and the q/k/v vectors (64, or 3 boxes in the figure)

Parsing Encoder: Multi-Head Attention and FFN

1) This is our input sentence*

2) We embed each word*

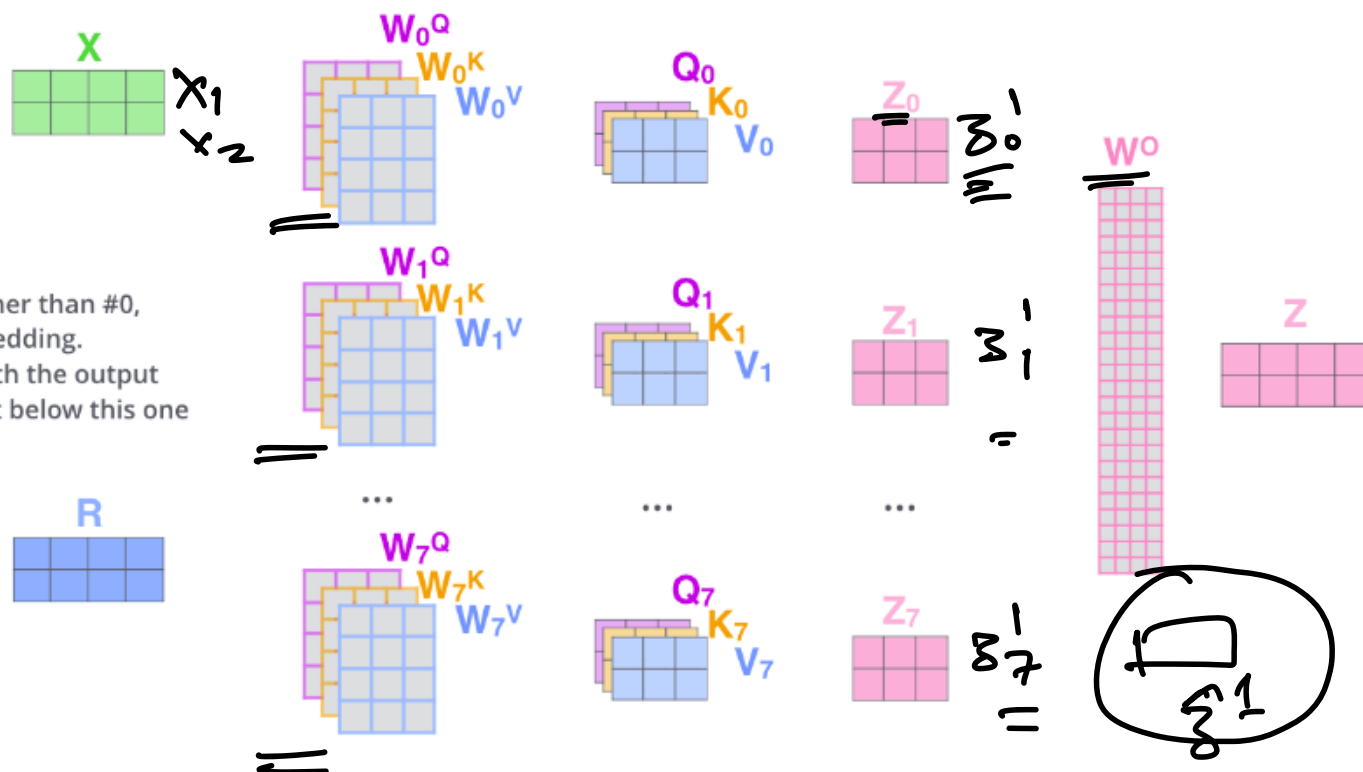
3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting $Q/K/V$ matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



Miniproject 1 Credit 4 Grading

1. Team Submission for Part 1, Part 2, Part 3, Part 4

— Have a paragraph that describes each team member's Contribution

2. Bonus of 10% grade if you are in top 10 on leader board

3. Bonus on the ICE coding that we had last week

ICE #0: Self-attention Exercise

Let's go through a self-attention python calculation exercise to understand it better. Let $x = [[1, 2, 3, -1], [3, -4, -7, 5]]$ be the input token embeddings. In the first layer of the encoder of the transformer, the weight matrices are given by $W^Q = [[-1, 2, 0], [2, 3, -5], [1, 0, 0], [-3, 1, 2]]$, $W^K = [[1, 2, 3], [2, 4, 3], [3, 0, 3], [-1, 5, 2]]$, $W^V = [[-1, -2, 3], [2, -4, 0], [0, 0, 1], [1, 0, -7]]$. Compute the soft-max similar to what we did in the previous walk-through. You can use python matrix multiplication (e.g. numpy) to arrive at the solution. Question is which token (token 1 or token 2) does token 2 place more attention on?

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?

I love to hike

I **mask** to hike

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)

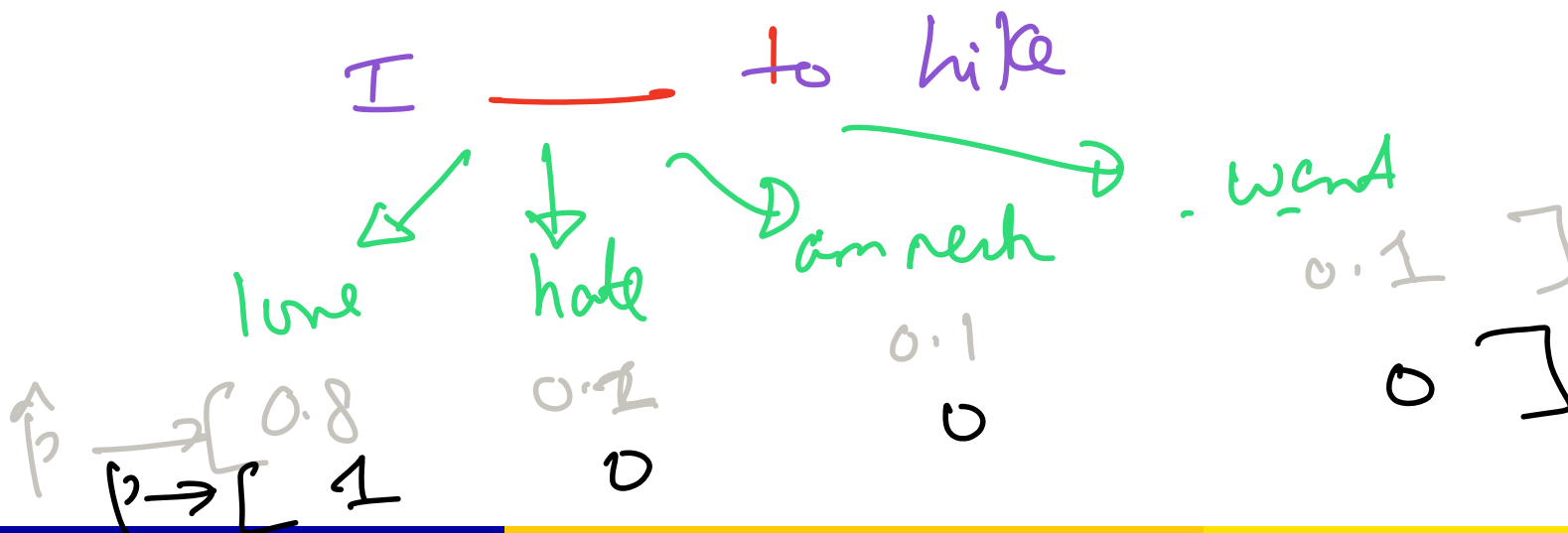
Loss Function for MLM mimicks which type of classic ML model?

Loss Function for Masked Language Model (MLM)

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

Cross-Entropy

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



Loss Function for Masked Language Model (MLM)

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

Cross-Entropy

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

ICE: What is the loss function for Binary Classification?

↓
Binary Cross Entropy

BERT - Bi-directional Encoders from Transformers

| System | MNLI-(m/mm) 392k | QQP 363k | QNLI 108k | SST-2 67k | CoLA 8.5k | STS-B 5.7k | MRPC 3.5k | RTE 2.5k | Average |
|-----------------------|---------------------|-------------|--------------|--------------|--------------|---------------|--------------|-------------|-------------|
| 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 |
| BERT _{BASE} | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| BERT _{LARGE} | 86.7/85.9 | 72.1 | 92.7 | 94.9 | 60.5 | 86.5 | 89.3 | 70.1 | 82.1 |

Decoding
↓
Encoding

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.

Sentence BERT - Siamese BERT architecture

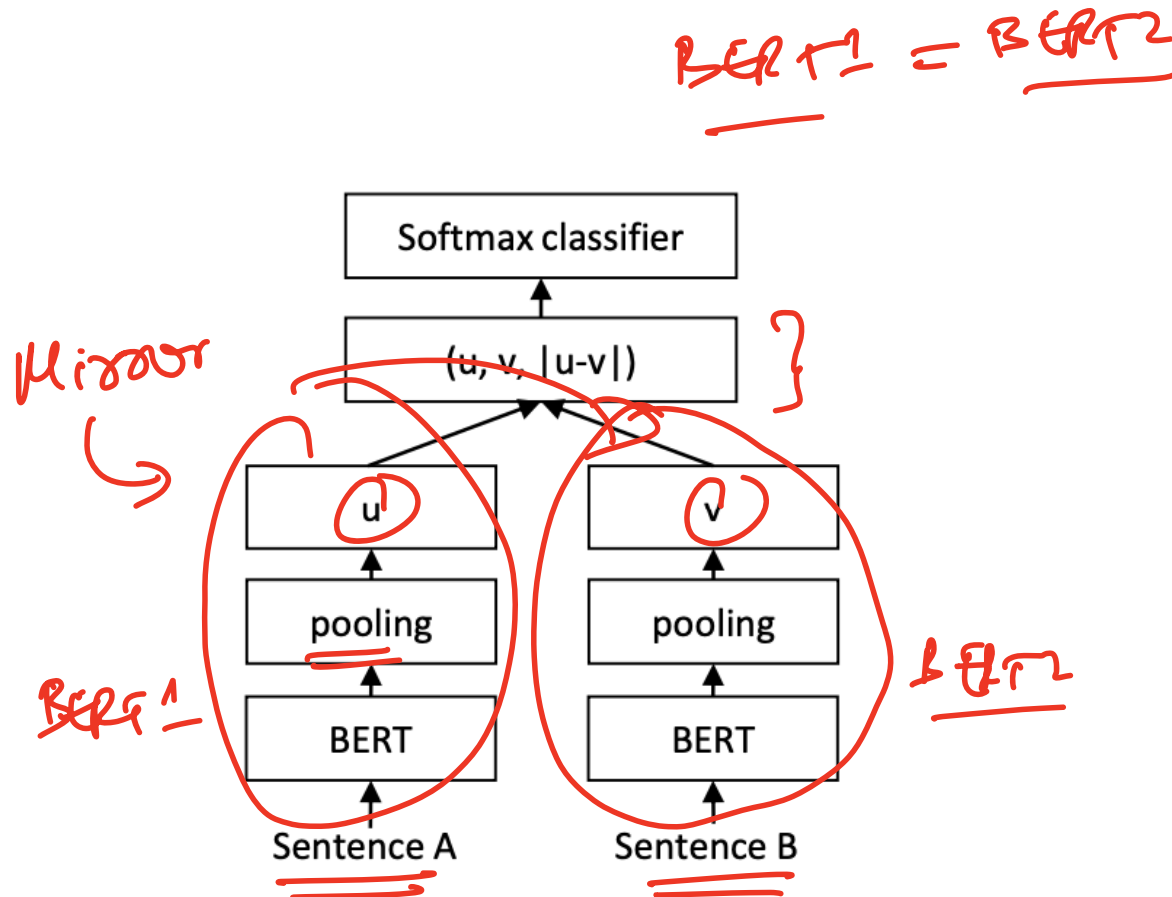


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

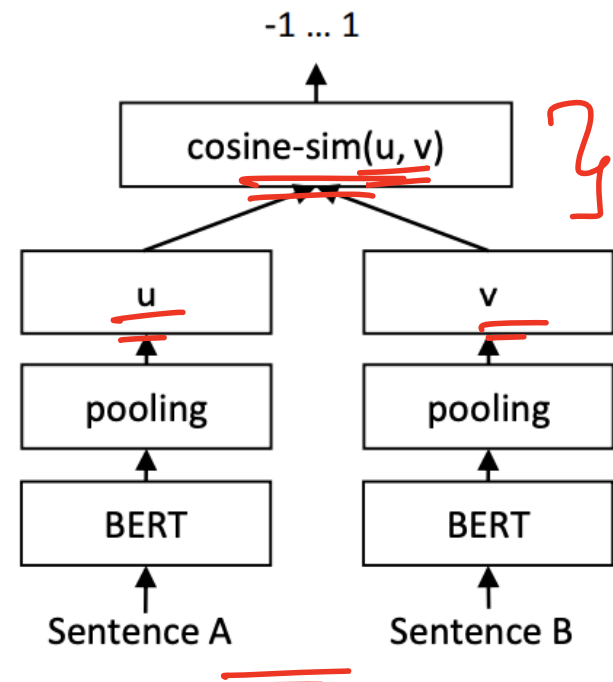


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Loss Function for SBERT

Pooling Strategy for SBERT


| | NLI | STSb |
|--------------------------|--------------|--------------|
| <i>Pooling Strategy</i> | | |
| <u>MEAN</u> | 80.78 | 87.44 |
| MAX | 79.07 | 69.92 |
| <u>CLS</u> | <u>79.80</u> | 86.62 |
| <i>Concatenation</i> | | |
| (u, v) | 66.04 | - |
| $(u - v)$ | 69.78 | - |
| $(u * v)$ | 70.54 | - |
| $(u - v , u * v)$ | 78.37 | - |
| $(u, v, u * v)$ | 77.44 | - |
| $(u, v, u - v)$ | 80.78 | - |
| $(u, v, u - v , u * v)$ | 80.44 | - |

Start token

3

Table 6: SBERT trained on NLI data with the classification objective function, on the STS benchmark (STSb) with the regression objective function. Configurations are evaluated on the development set of the STSb using cosine-similarity and Spearman's rank correlation. For the concatenation methods, we only report scores with MEAN pooling strategy.

Sentence BERT Cosine Similarity Results



| Model | STS12 | STS13 | STS14 | STS15 | STS16 | STSb | SICK-R | Avg. |
|----------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Avg. GloVe embeddings | <u>55.14</u> | 70.66 | 59.73 | 68.25 | 63.66 | 58.02 | 53.76 | 61.32 |
| Avg. BERT embeddings | <u>38.78</u> | 57.98 | 57.98 | 63.15 | 61.06 | 46.35 | 58.40 | 54.81 |
| BERT CLS-vector | <u>20.16</u> | <u>30.01</u> | 20.09 | <u>36.88</u> | 38.08 | 16.50 | 42.63 | 29.19 |
| InferSent - Glove | 52.86 | 66.75 | <u>62.15</u> | <u>72.77</u> | 66.87 | 68.03 | 65.65 | 65.01 |
| Universal Sentence Encoder | 64.49 | 67.80 | 64.61 | 76.83 | 73.18 | 74.92 | 76.69 | 71.22 |
| SBERT-NLI-base | 70.97 | 76.53 | 73.19 | 79.09 | 74.30 | 77.03 | 72.91 | 74.89 |
| SBERT-NLI-large | 72.27 | 78.46 | 74.90 | <u>80.99</u> | 76.25 | 79.23 | 73.75 | 76.55 |
| SRoBERTa-NLI-base | 71.54 | 72.49 | 70.80 | 78.74 | 73.69 | 77.77 | 74.46 | 74.21 |
| SRoBERTa-NLI-large | 74.53 | 77.00 | 73.18 | 81.85 | 76.82 | 79.10 | 74.29 | 76.68 |

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

SentEval DataSets

- **MR**: Sentiment prediction for movie reviews snippets on a five star scale (Pang and Lee, 2005).
- **CR**: Sentiment prediction of customer product reviews (Hu and Liu, 2004).
- **SUBJ**: Subjectivity prediction of sentences from movie reviews and plot summaries (Pang and Lee, 2004).
- **MPQA**: Phrase level opinion polarity classification from newswire (Wiebe et al., 2005).
- **SST**: Stanford Sentiment Treebank with binary labels (Socher et al., 2013).
- **TREC**: Fine grained question-type classification from TREC (Li and Roth, 2002).
- **MRPC**: Microsoft Research Paraphrase Corpus from parallel news sources (Dolan et al., 2004).

Sentence BERT on SentEval Results

| Model | MR | CR | SUBJ | MPQA | SST | TREC | MRPC | Avg. |
|----------------------------|---------------------|---------------------|---------------------|------------------|--------------|-------------|--------------|--------------|
| Avg. GloVe embeddings | 77.25 | 78.30 | 91.17 | 87.85 | 80.18 | 83.0 | 72.87 | 81.52 |
| Avg. fast-text embeddings | 77.96 | 79.23 | 91.68 | 87.81 | 82.15 | 83.6 | 74.49 | 82.42 |
| Avg. BERT embeddings | <u>78.66</u> | <u>86.25</u> | 94.37 | 88.66 | 84.40 | 92.8 | 69.45 | 84.94 |
| BERT CLS-vector | 78.68 | <u>84.85</u> | 94.21 | 88.23 | 84.13 | 91.4 | 71.13 | 84.66 |
| InferSent - GloVe | 81.57 | 86.54 | 92.50 | 90.38 | 84.18 | 88.2 | 75.77 | 85.59 |
| Universal Sentence Encoder | 80.09 | 85.19 | 93.98 | 86.70 | 86.38 | 93.2 | 70.14 | 85.10 |
| SBERT-NLI-base | 83.64 | 89.43 | 94.39 | 89.86 | 88.96 | 89.6 | 76.00 | 87.41 |
| SBERT-NLI-large | <u>84.88</u> | <u>90.07</u> | <u>94.52</u> | 90.33 | 90.66 | 87.4 | 75.94 | 87.69 |

Table 5: Evaluation of SBERT sentence embeddings using the SentEval toolkit. SentEval evaluates sentence embeddings on different sentence classification tasks by training a logistic regression classifier using the sentence embeddings as features. Scores are based on a 10-fold cross-validation.

ICE #1

Let's say we want to automatically convert a **Natural Language Query** to a **SQL** query. E.g. "Which quarter in the past 5 years had the most amount of sales for fashion products" to "SELECT ... FROM ... WHERE ...". What kind of deep learning architecture would support this problem?

- ① SBERT
- ② LSTM to LSTM sequence model
- ③ GPT-2
- ④ Feed Forward Neural Network

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

- ① **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: bert-base-uncased is one such pre-trained model that can be loaded through Hugging Face Transformers Library

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

- ① **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: bert-base-uncased is one such pre-trained model that can be loaded through Hugging Face Transformers Library
- ② **Extract output from pre-training:** How do you want to use the output from pre-training going into *fine-tuning*? a) Extract embedding from the first token, CLS b) Average embeddings of all tokens as a starting point (mean pooling).

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

- 1 **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: bert-base-uncased is one such pre-trained model that can be loaded through Hugging Face Transformers Library
- 2 **Extract output from pre-training:** How do you want to use the output from pre-training going into *fine-tuning*? a) Extract embedding from the first token, CLS b) Average embeddings of all tokens as a starting point (mean pooling).
- 3 **Add fine-tuning layers:** Add fine-tuning layers on top of the pre-trained layers. Example, starting with the pooled embeddings, construct one or more dense layers (Feed-Forward NN style) to extract finer representations of the input. Add the output layer and its activation (typically softmax for classification tasks).

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

- ① **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: bert-base-uncased is one such pre-trained model that can be loaded through Hugging Face Transformers Library
- ② **Extract output from pre-training:** How do you want to use the output from pre-training going into *fine-tuning*? a) Extract embedding from the first token, CLS b) Average embeddings of all tokens as a starting point (mean pooling).
- ③ **Add fine-tuning layers:** Add fine-tuning layers on top of the pre-trained layers. Example, starting with the pooled embeddings, construct one or more dense layers (Feed-Forward NN style) to extract finer representations of the input. Add the output layer and its activation (typically softmax for classification tasks).
- ④ **Set training schedule, hyper-parameters, etc:** Set up optimizer (e.g. ADAM), hyper-parameters, training schedule, etc for training.

ICE #2

BERT Embeddings and Emotion Detection

Let's say you want to do emotion detection by fine-tuning BERT (Encoding Transformer) on a data set. One of the outputs of the *BERT pre-trained model* for a given input is the *last-hidden-state*. This includes an embedding for every token that was passed into BERT.

Let's say you are going to start with the last hidden layer and use that as input for your *fine-tuned* model. This ICE is about the dimensionality of the inputs and outputs. Let's say you have sentences of the kind: "I am looking forward to today! It's going to be a big day" This sentence conveys excitement. There are 13 words in this input and using *word-piece tokenization*, you arrive at 20 sub-tokens as input into the BERT model. The last hidden layer includes an embedding for every single token. Let's say the embedding dimension for a token is 768.

ICE #2 continued

BERT Embeddings and Emotion Detection

There are 13 words in this input and using *word-piece tokenization*, you arrive at 20 sub-tokens as input into the BERT model. The last hidden layer includes an embedding for every single token. Let's say the embedding dimension for a token is 768. For the purpose of emotion detection - You can either use the *CLS* token (Start token) embedding (also called the pooled embedding) or you can take the average of the embeddings of the tokens in the last hidden layer of BERT. a) What's the dimension of the pooled BERT embedding of this particular input example b) What's the dimension of the CLS/Start token embedding in this example? c) what's the total dimension of the last hidden layer?

- ① 768, 15630 and 768
- ② 768, 768 and 768
- ③ 15360, 768 and 15630
- ④ 768, 768 and 15360



ICE #3

Why does pooling of the output need to be done for sequence classification (e.g. emotion detection)?

- ① Reduces the dimensionality
- ② Averages context from all the tokens
- ③ Computational concerns for training the fine-tuned model
- ④ All of the above

Application of SBERT Embeddings to Instacart Recommendations

Instacart Recommendations

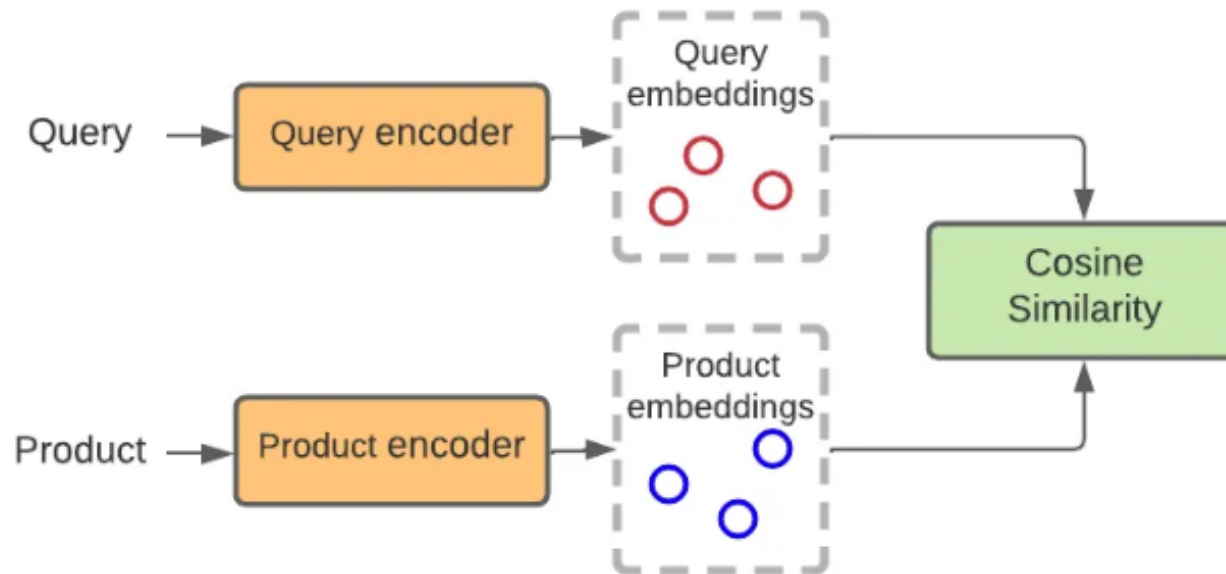
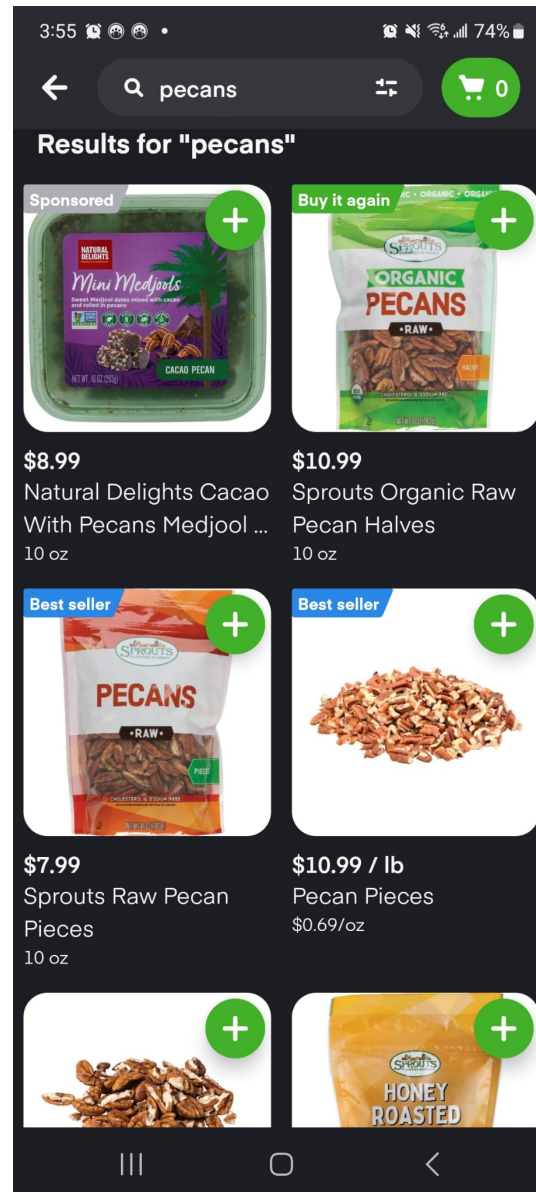


Figure 1. Conceptual diagram of a two-tower model

Positive Examples

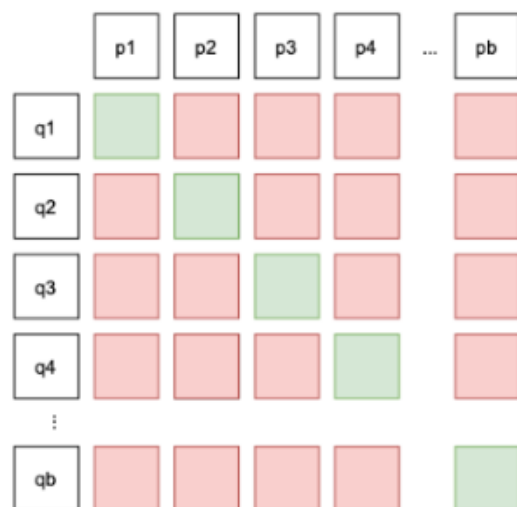


High-quality Positive Examples

| Converted Products for Search Query “Orange” |
|--|
| Navel Oranges |
| Clementines |
| Mandarins |
| ... |
| Bananas |
| ... |
| Strawberries |

Negative Examples

Vanilla In-batch Negative



In-batch Negative with Self-adversarial Re-weighting

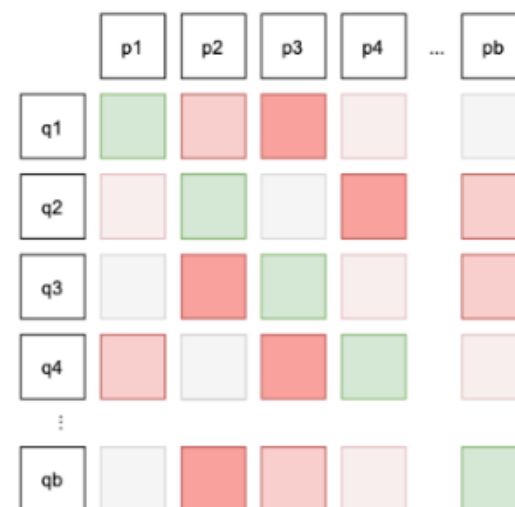


Figure 3. (Left) In the vanilla implementation of in-batch negative, all off-diagonal negative samples are given the same weight. (Right) In our implementation with self-adversarial re-weighting, harder examples are given more weight (darker color), making the task more challenging for the model.

Model Training Architecture

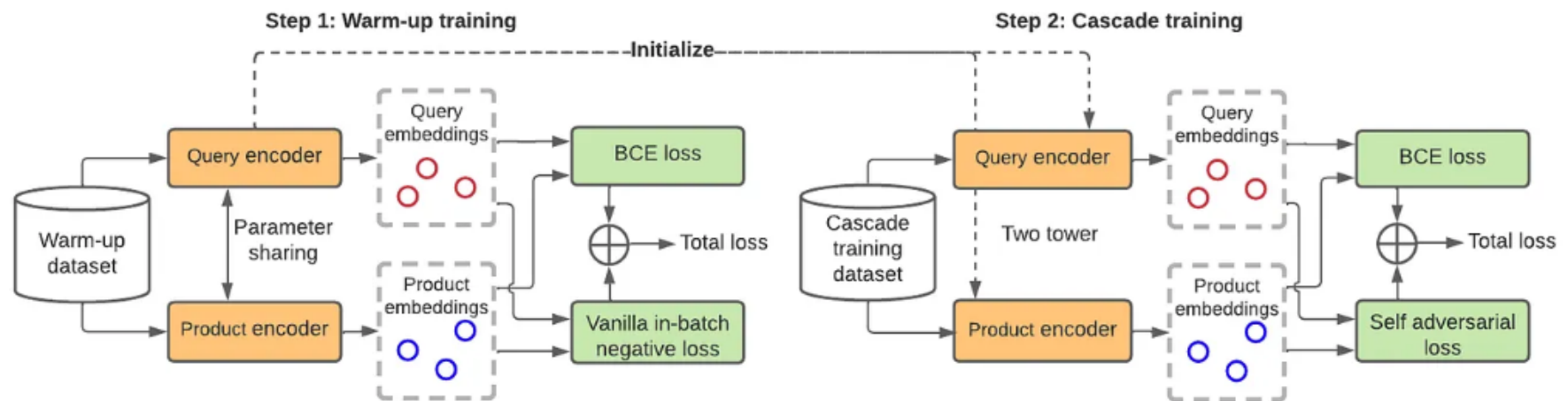


Figure 4. Two-step cascade training for ITEMS.

System Design

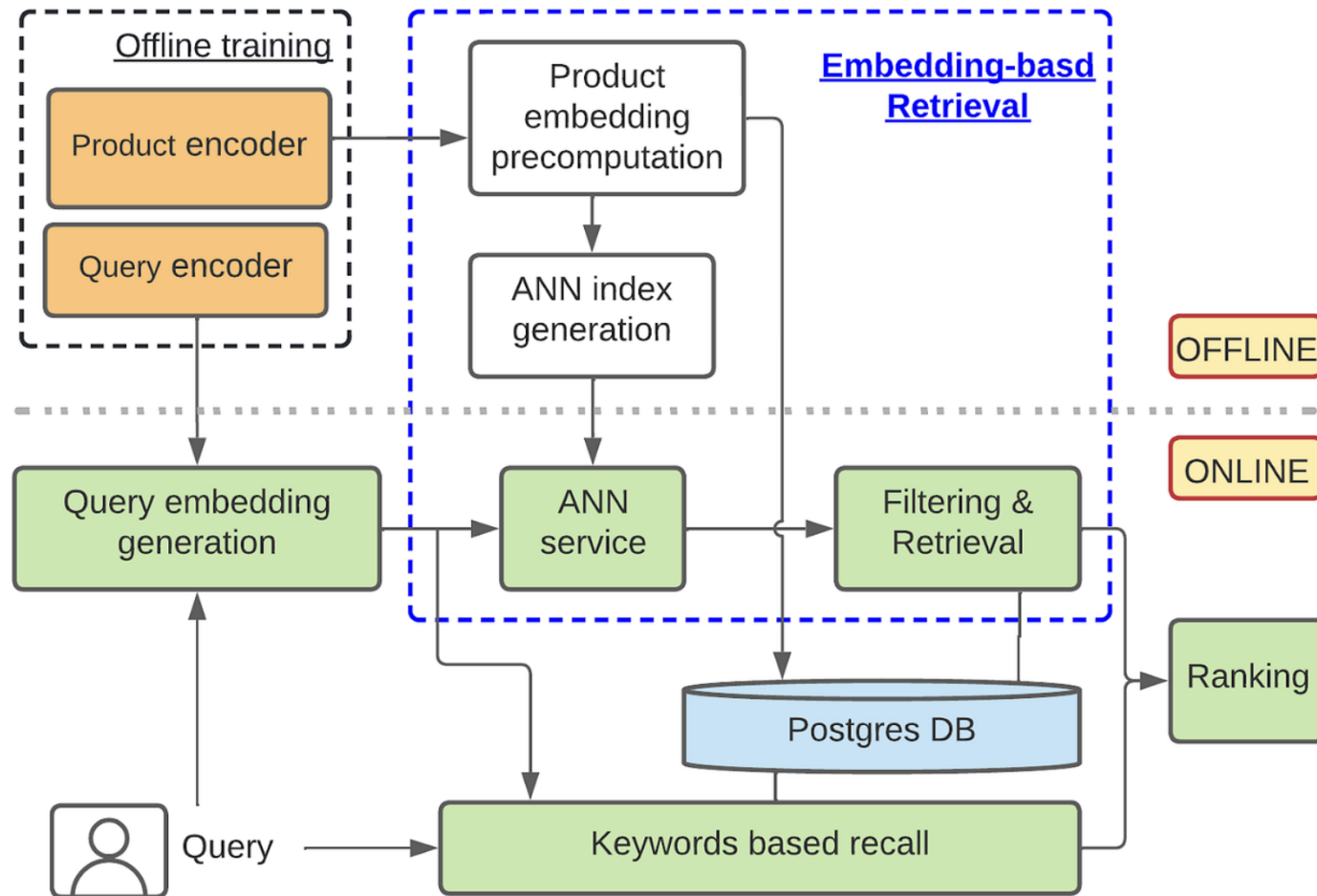


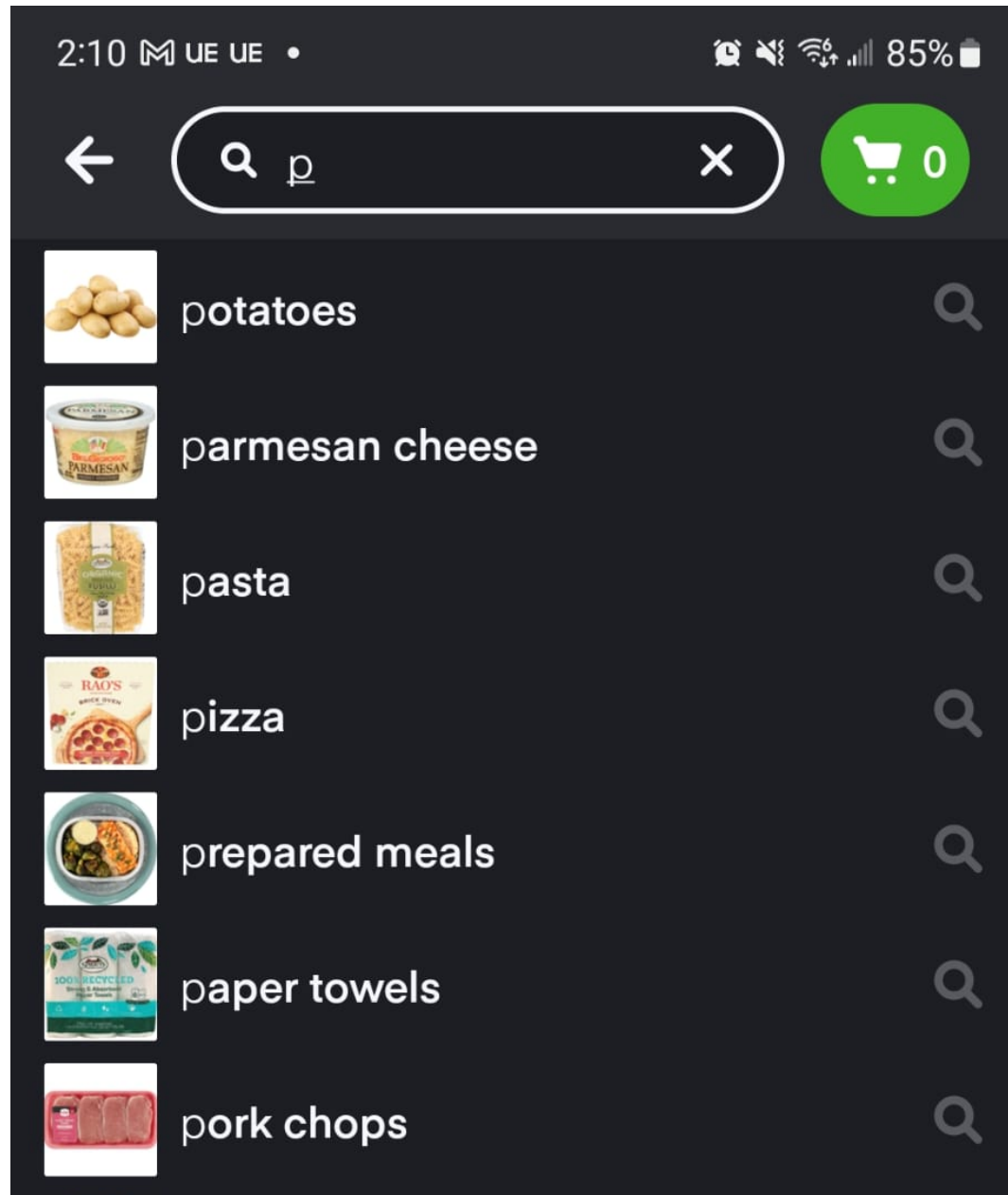
Figure 7. ITEMS system architecture.

Breakouts Time #1

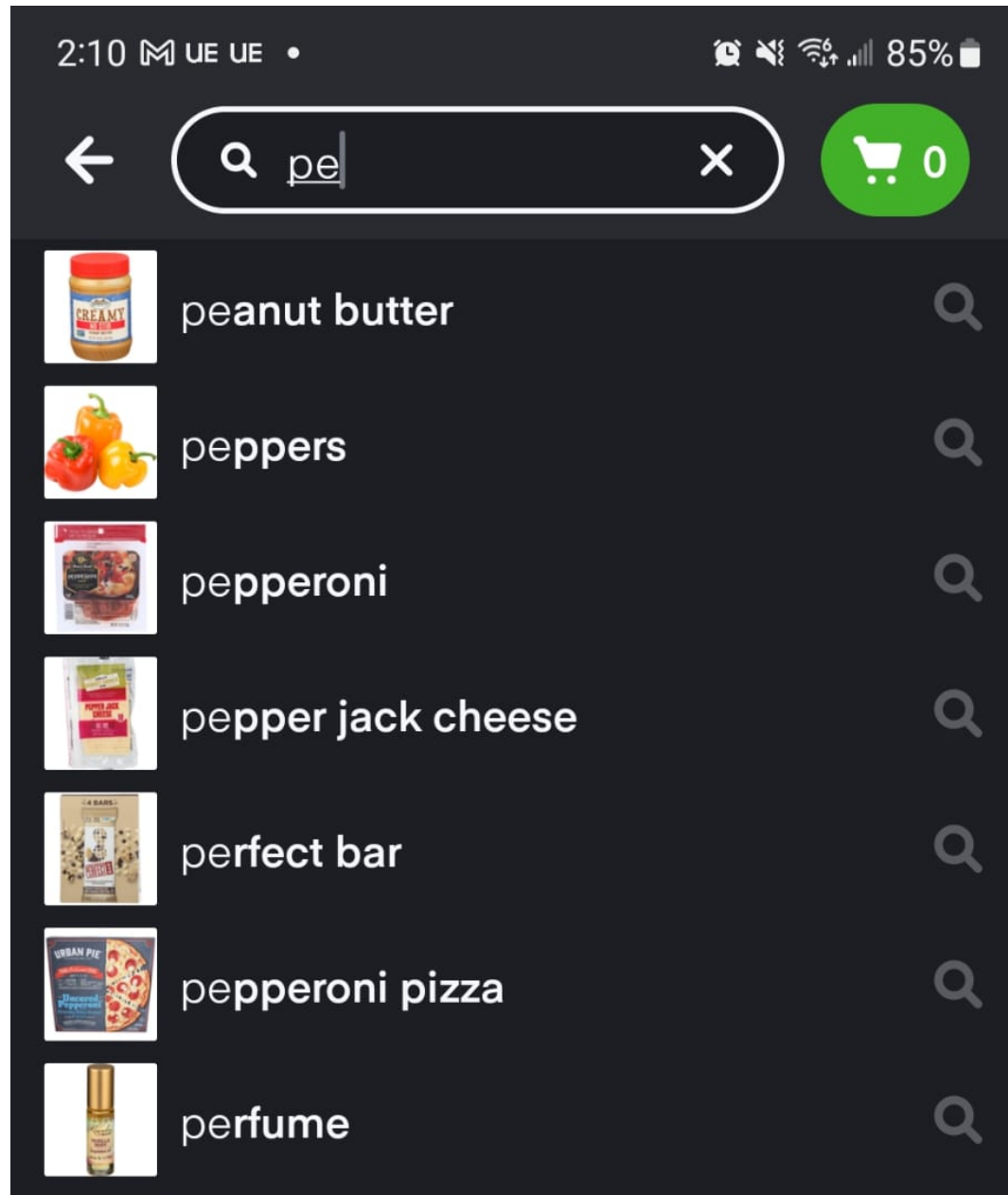
Auto-complete — 5 mins

Let's say you are tasked with building an in-email auto-completion application, which can help complete partial sentences into full sentences through suggestions (auto-complete). How would you use what we have learned so far to model this? What architecture would you use? What would be your data? And what are some pitfalls or painpoints your model should address?

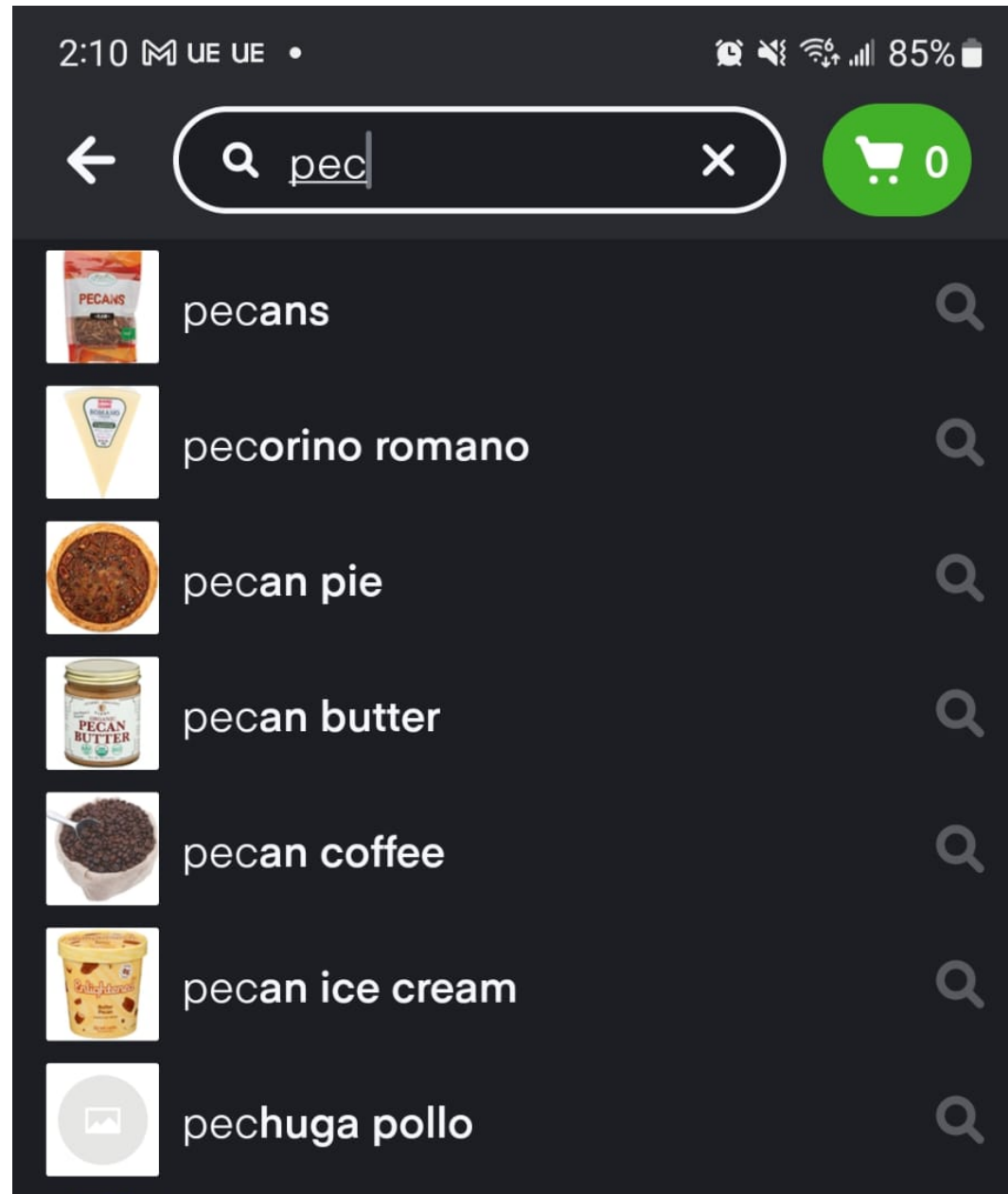
Instacart Auto-Complete and Search Relevance



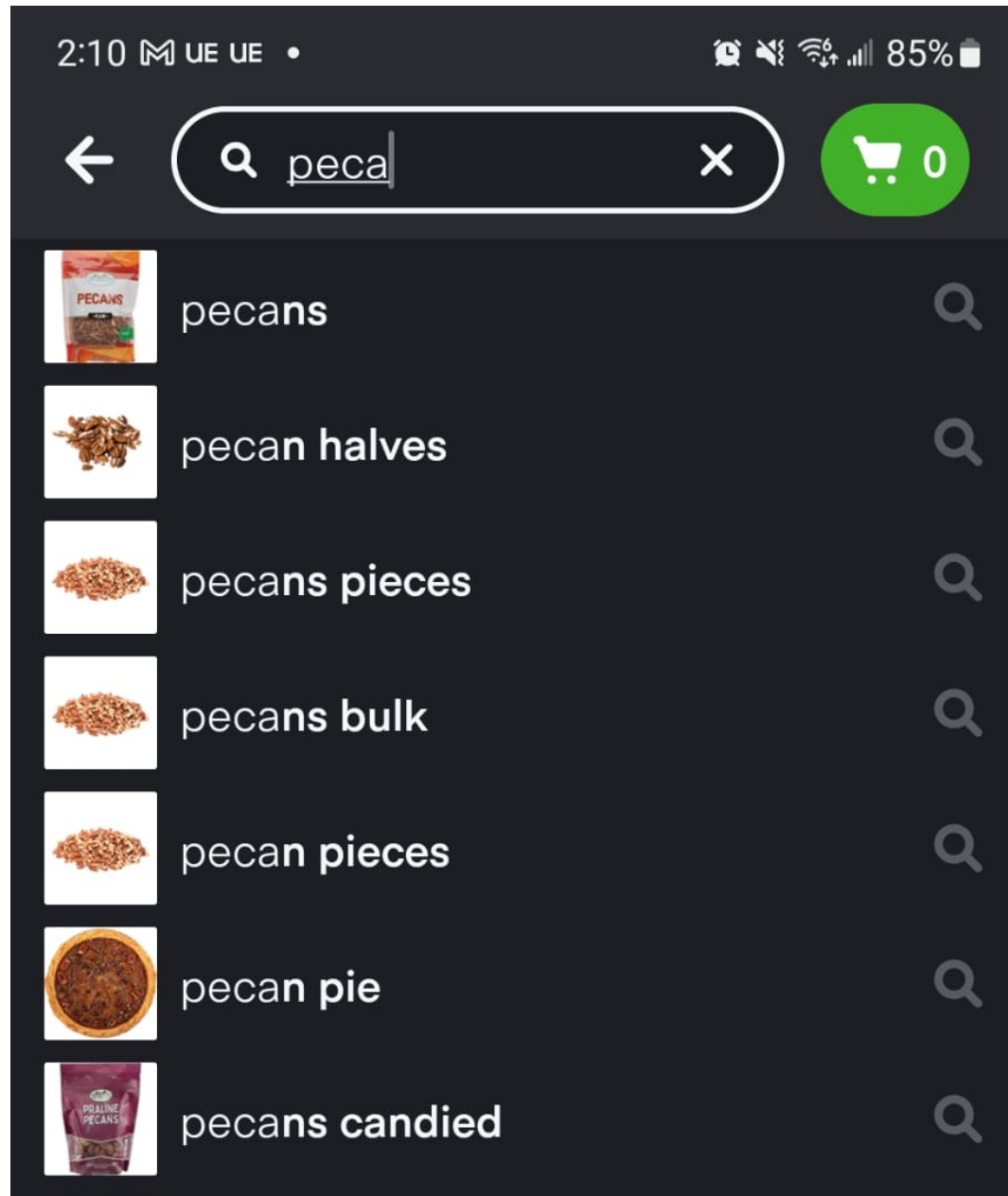
Instacart Auto-Complete



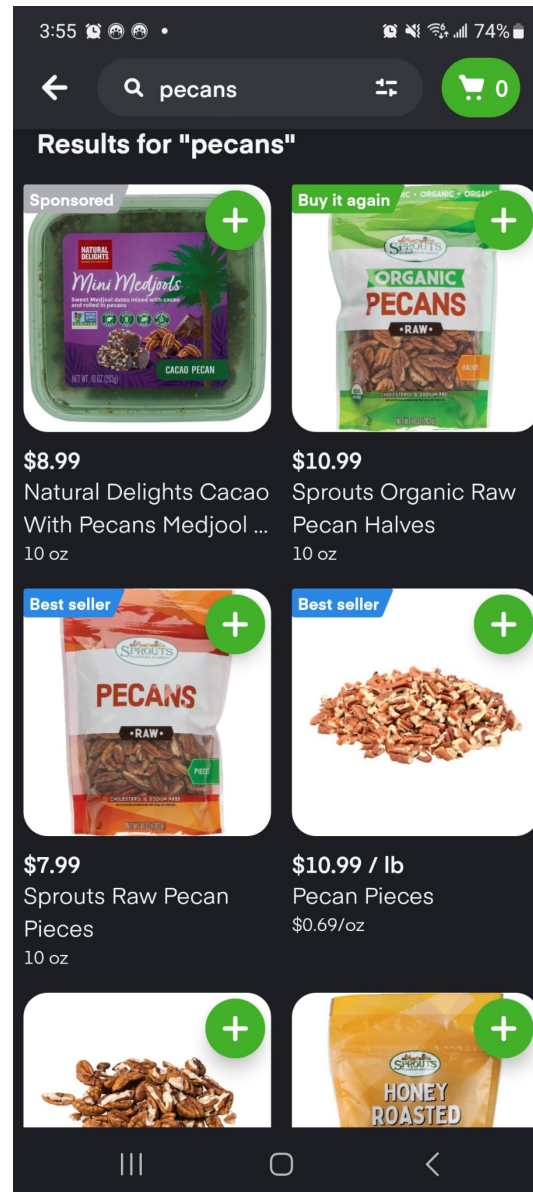
Instacart Auto-Complete



Instacart Auto-Complete



Instacart Auto-Complete and Search Results



Instacart Diversifying Auto-Complete

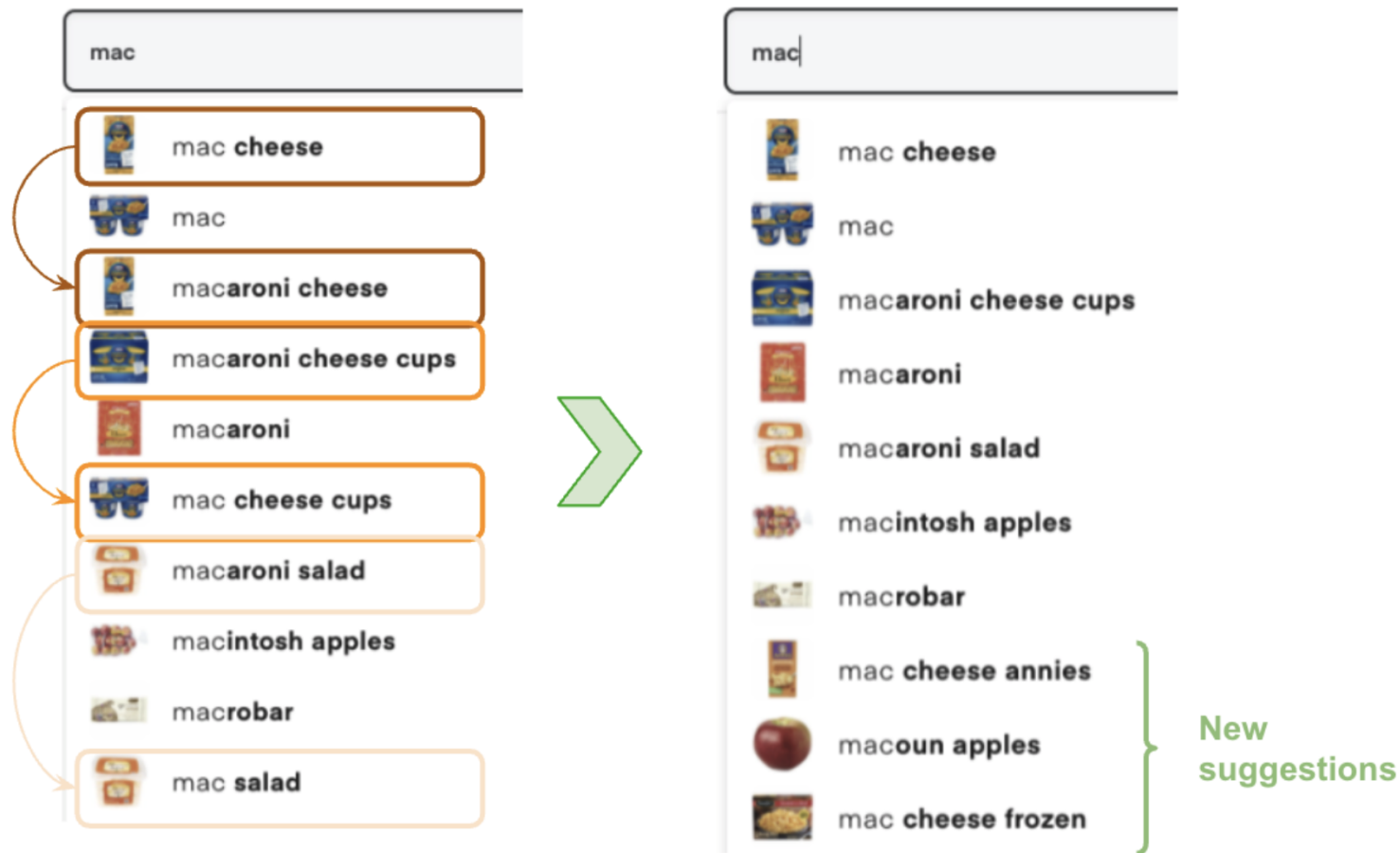


Figure 9. Autocomplete when a customer searches for “mac”, before (left) and after (right) semantic deduplication.