EEP 596: Adv Intro ML || Lecture 19

Dr. Karthik Mohan

Univ. of Washington, Seattle

March 10, 2023

todo-list

1. Transformers arch - Canva diagram - Have students answer
2. Encoder-Encoder Architecture and Two-Tower Architecture
3. Ask students to engage on this
4. How to get sentence encoding from bert?
5. Sentence BERT - Overcome limitations of BERT
6. Instacart Transformers Use-case
7. MRPC notebook finish if time permits

Today

- Transformers Architectures Recap
- Encoder-Encoder/Siamese Networks and Two-Tower Architecture
- Sentence Transformer - Sentence BERT or SBERT
- Instacart Recommendations using Transformers

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

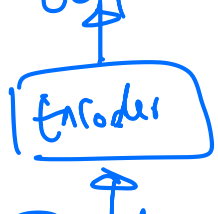
A methodology for fine-tuning transformers for classification tasks

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.

Transformers Use-Cases Over-view

1. only Encoder

output → Down-stream task



Input

Applns:-

a) Summarizing a sentence

b) Sentiment Analysis

c) Token classification

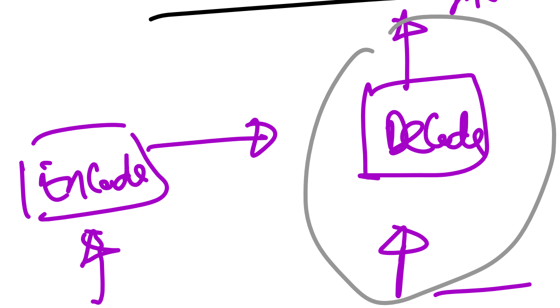
POS (parts of speech)

Entity Recognition (Name, Place)

d) Question-Answering (Extractive)

e) Vision Encoding:- ViTransformer

2. Encoder-Decoder Archs



Input

output Generation

Applns:-

a) Chat-GPT!

b) Free-form Q&A

c) Translate English to French

3. Encoder - Encoder Architecture

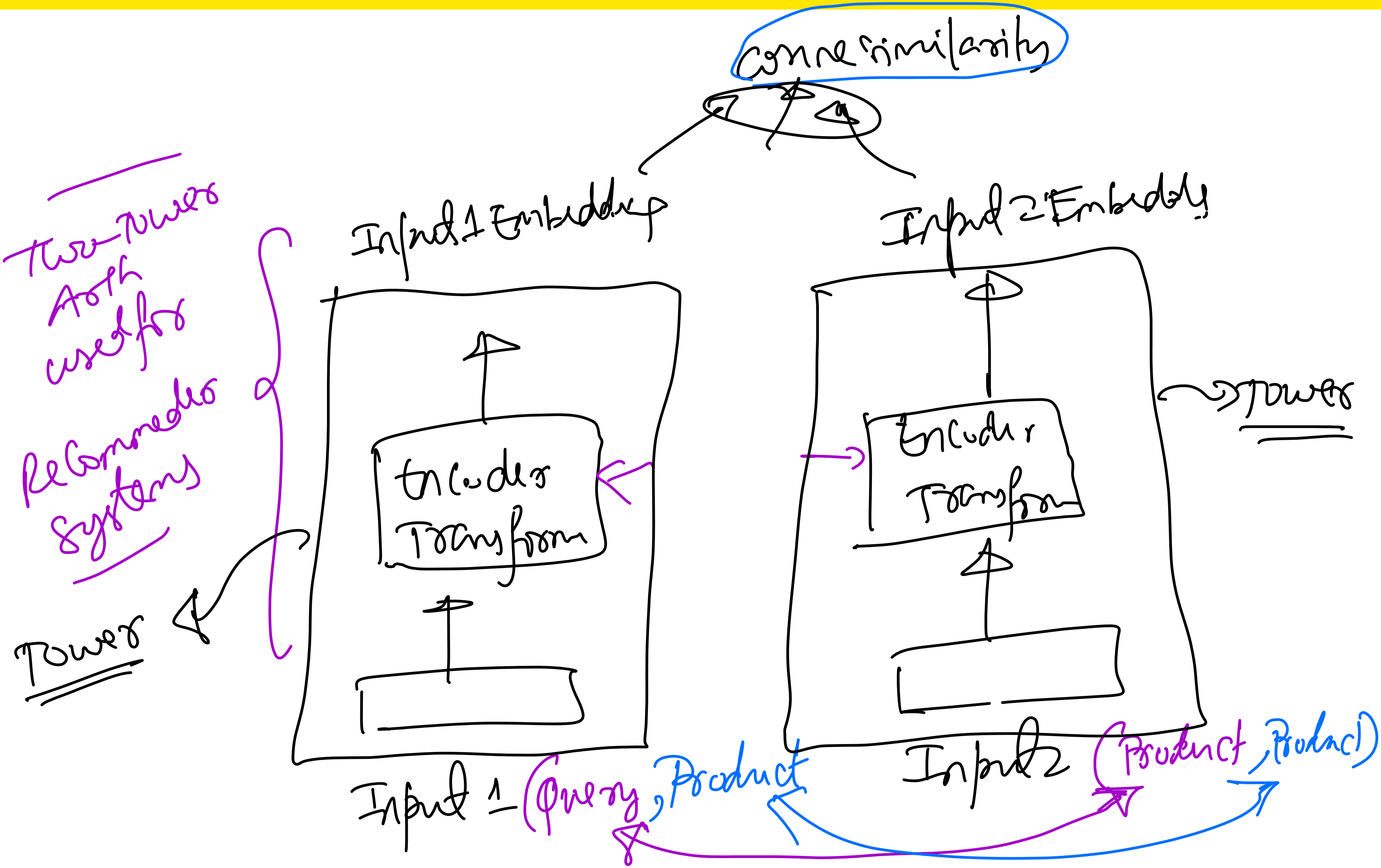
- Two Tower Architecture
- Siamese — " —

Applns: -

a) Search for a product similar to a given product

b) Return top K products for a query!

Use-Cases for Two Tower Architecture



Two-Tower Architecture

Sentence BERT

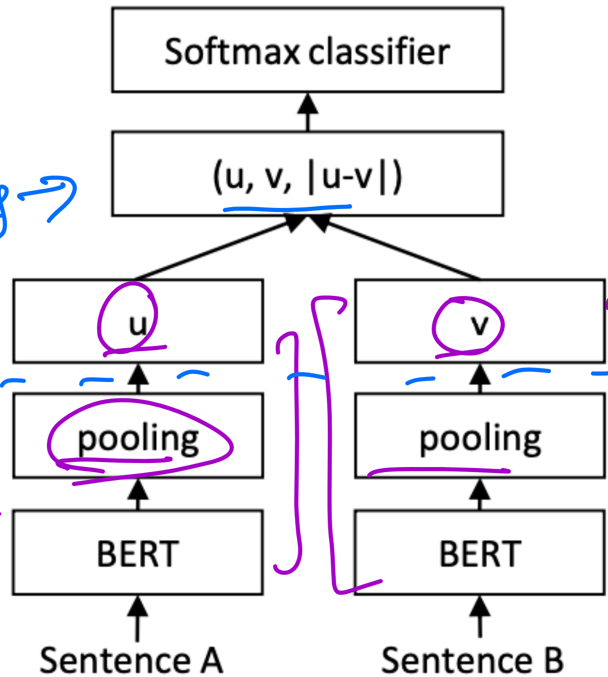


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

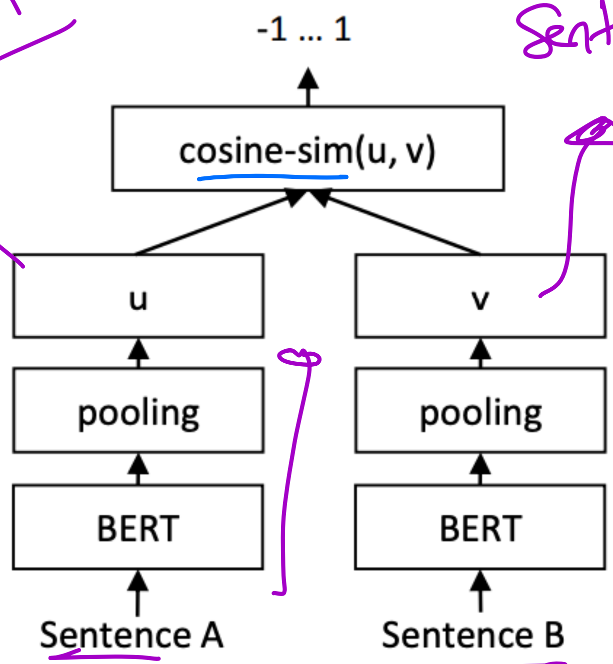


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Pooling Strategy for SBERT

	NLI	STSb
Pooling Strategy		
MEAN	**80.78**	**87.44**
MAX	79.07	69.92
CLS	79.80	86.62
Concatenation		
(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	-

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	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	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**	80.99	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 start 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	78.66	86.25	94.37	88.66	84.40	92.8	69.45	84.94
BERT CLS-vector	78.68	84.85	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	**84.88**	**90.07**	**94.52**	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.

Breakout

Given all the architecutures and knowledge of embeddings that we have discussed so far in class - Discuss your approaches for Kaggle 1 and Kaggle 2 tasks. Let's say you have an approach for Kaggle 1 contest. You have put in your submission for the leaderboard. Can you leverage your model for Kaggle 1 with add ons/creativity to do zero-shot learning in Kaggle 2? Remember, in Kaggle 2 - The input includes labels the model hasn't seen yet in training!

Instacart Recommendations

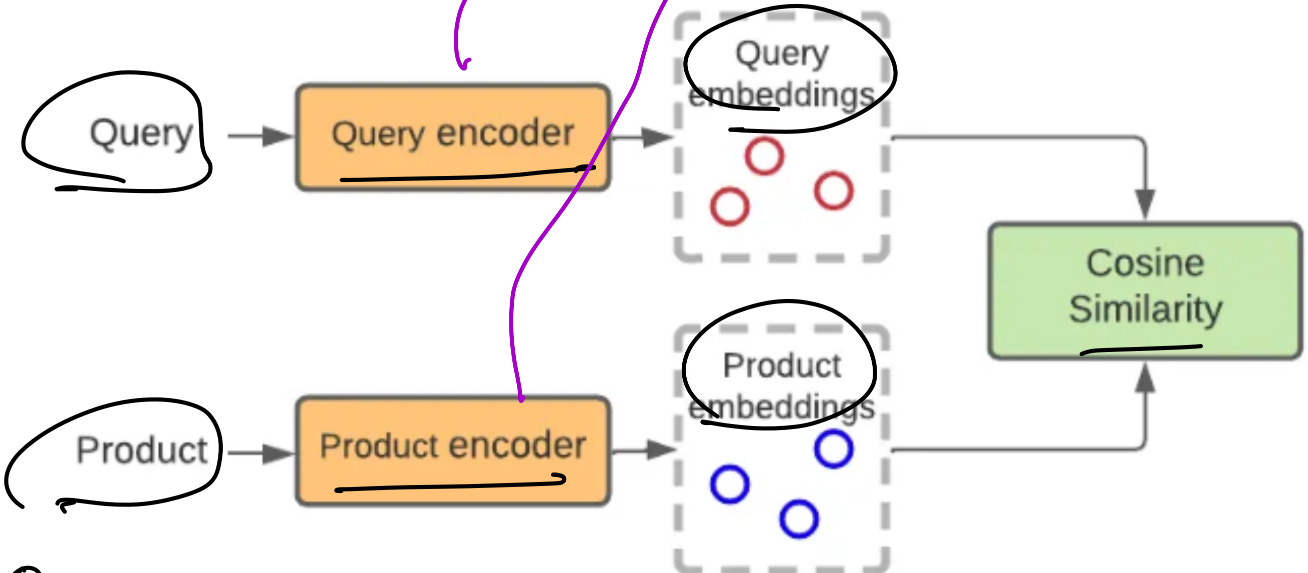
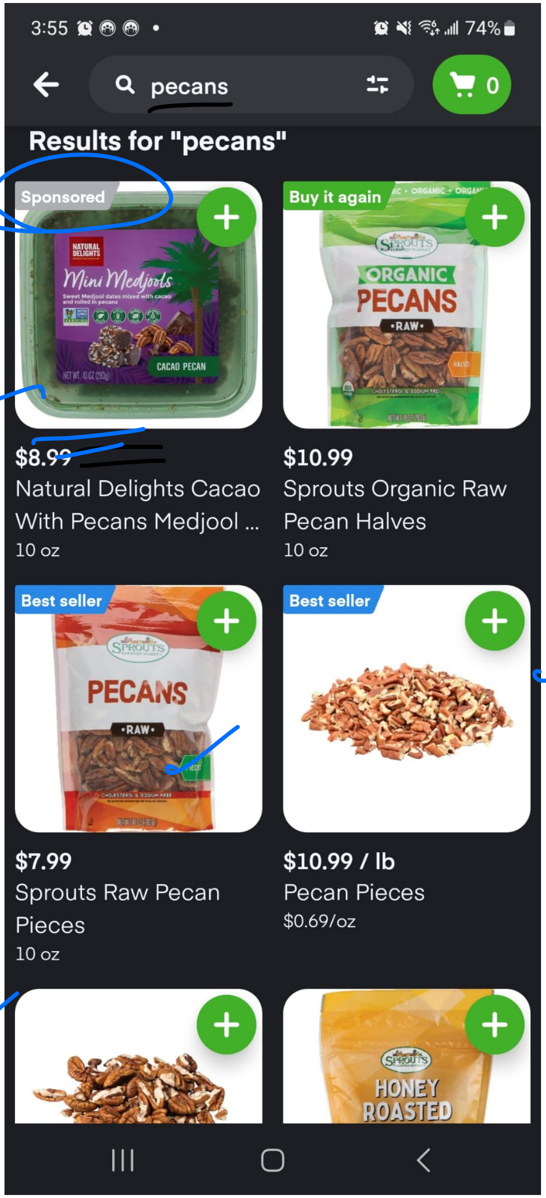


Figure 1. Conceptual diagram of a two-tower model

Positive Examples



Not a good product!

High-quality Positive Examples

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

Negative Examples

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

use Sentence Bert

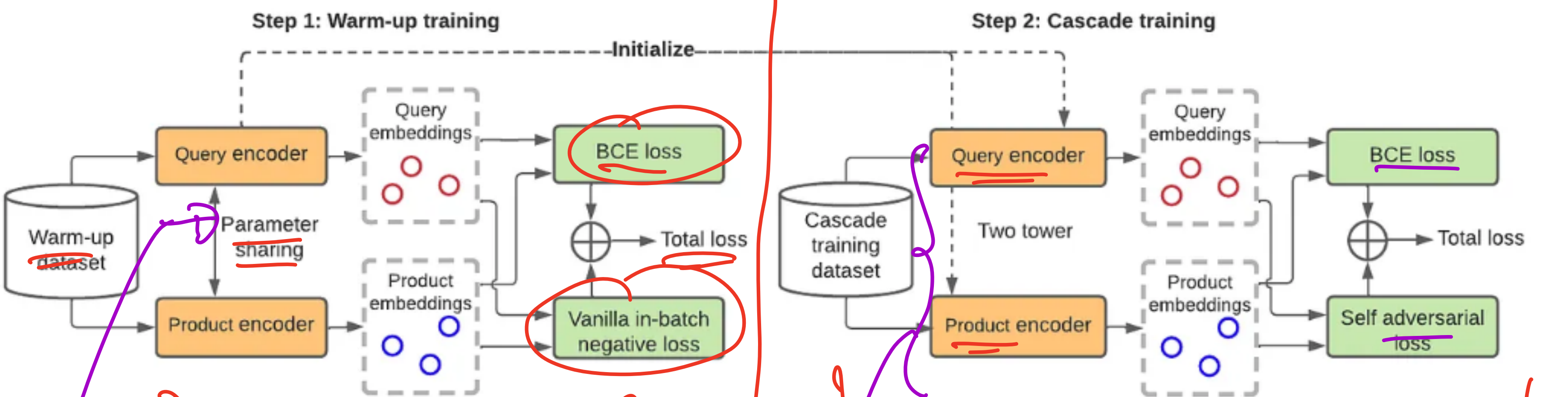


Figure 4. Two-step cascade training for ITEMS.

Siamese Two-tower

Non-Siamese Two-tower

System Design

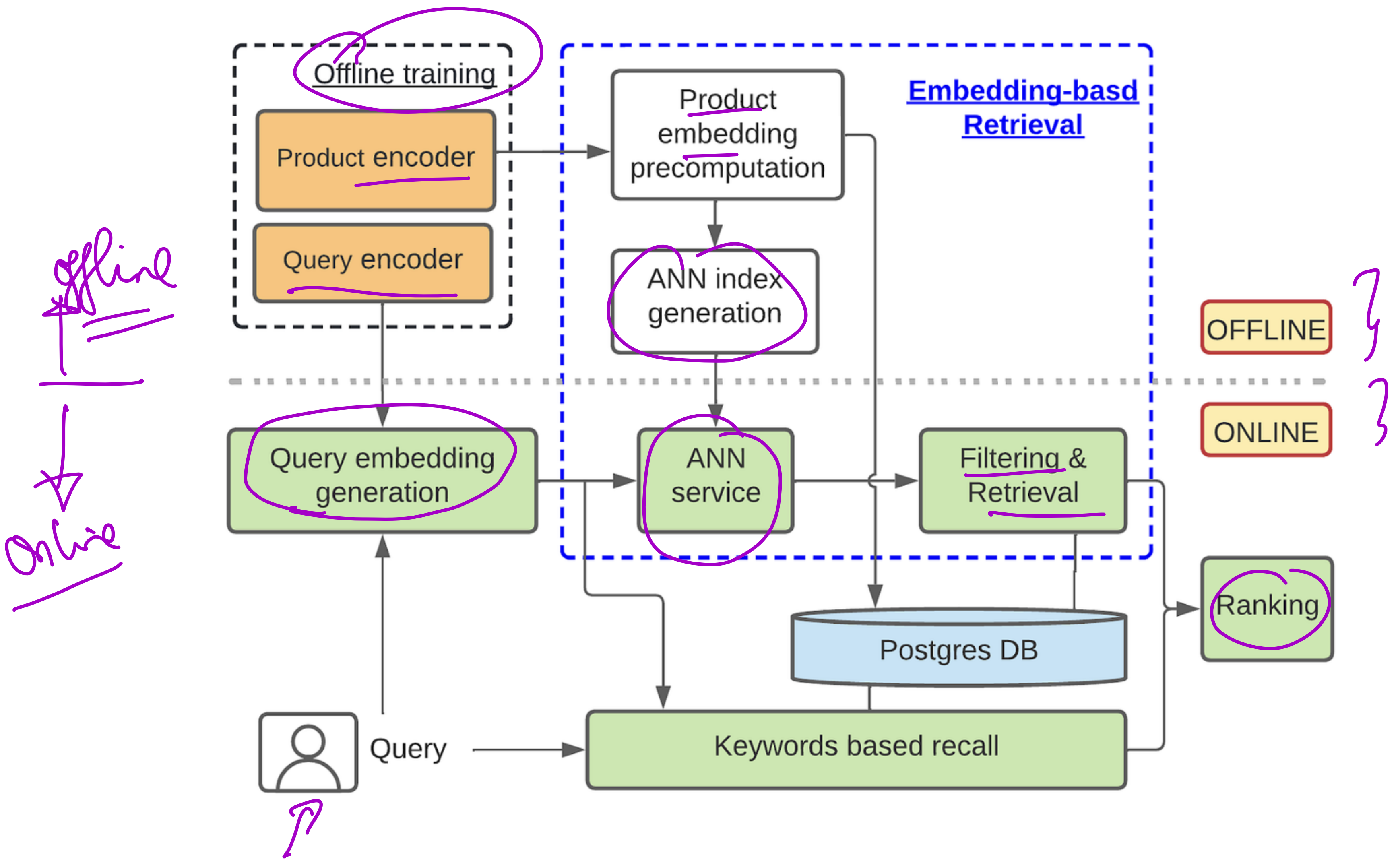
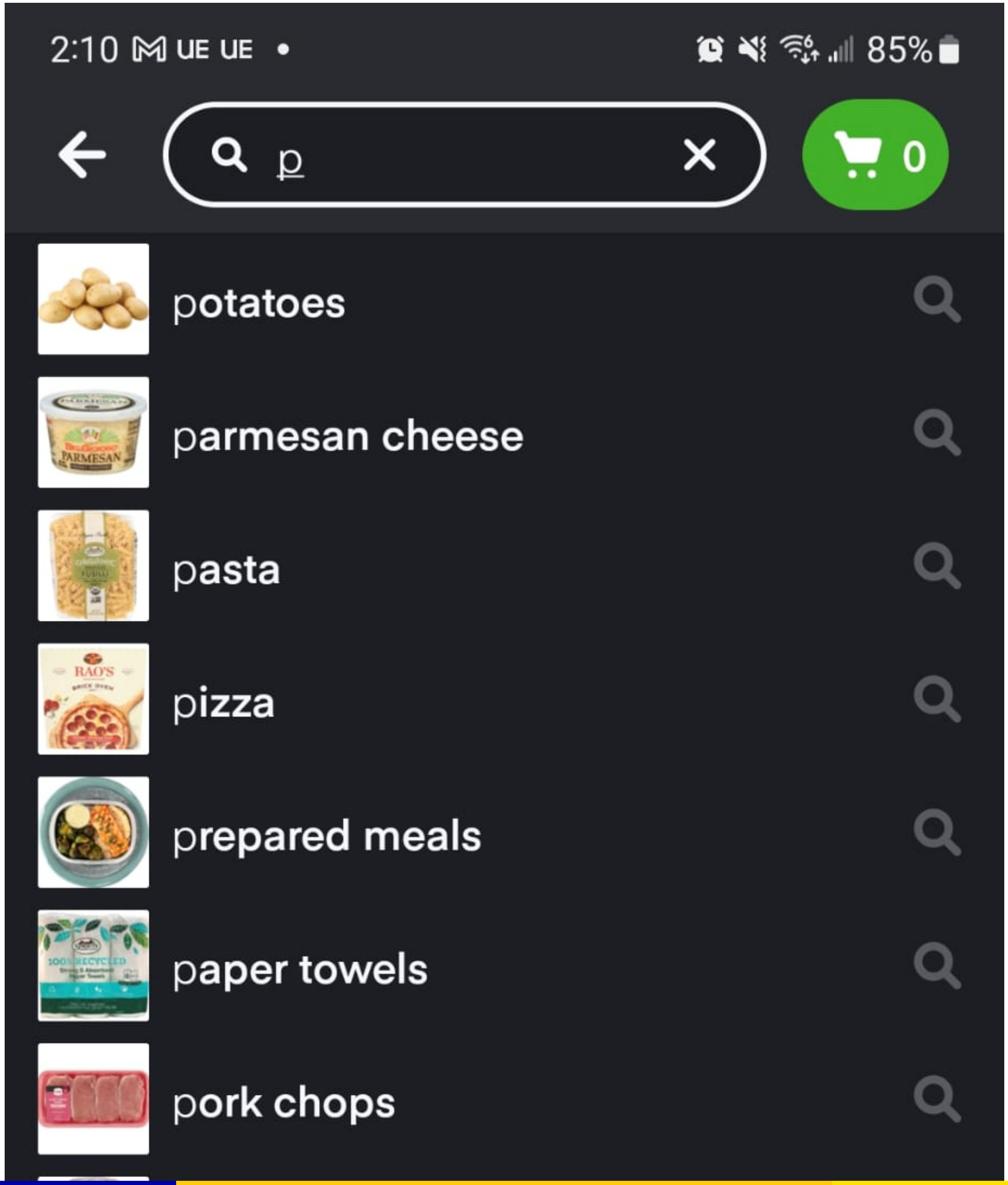
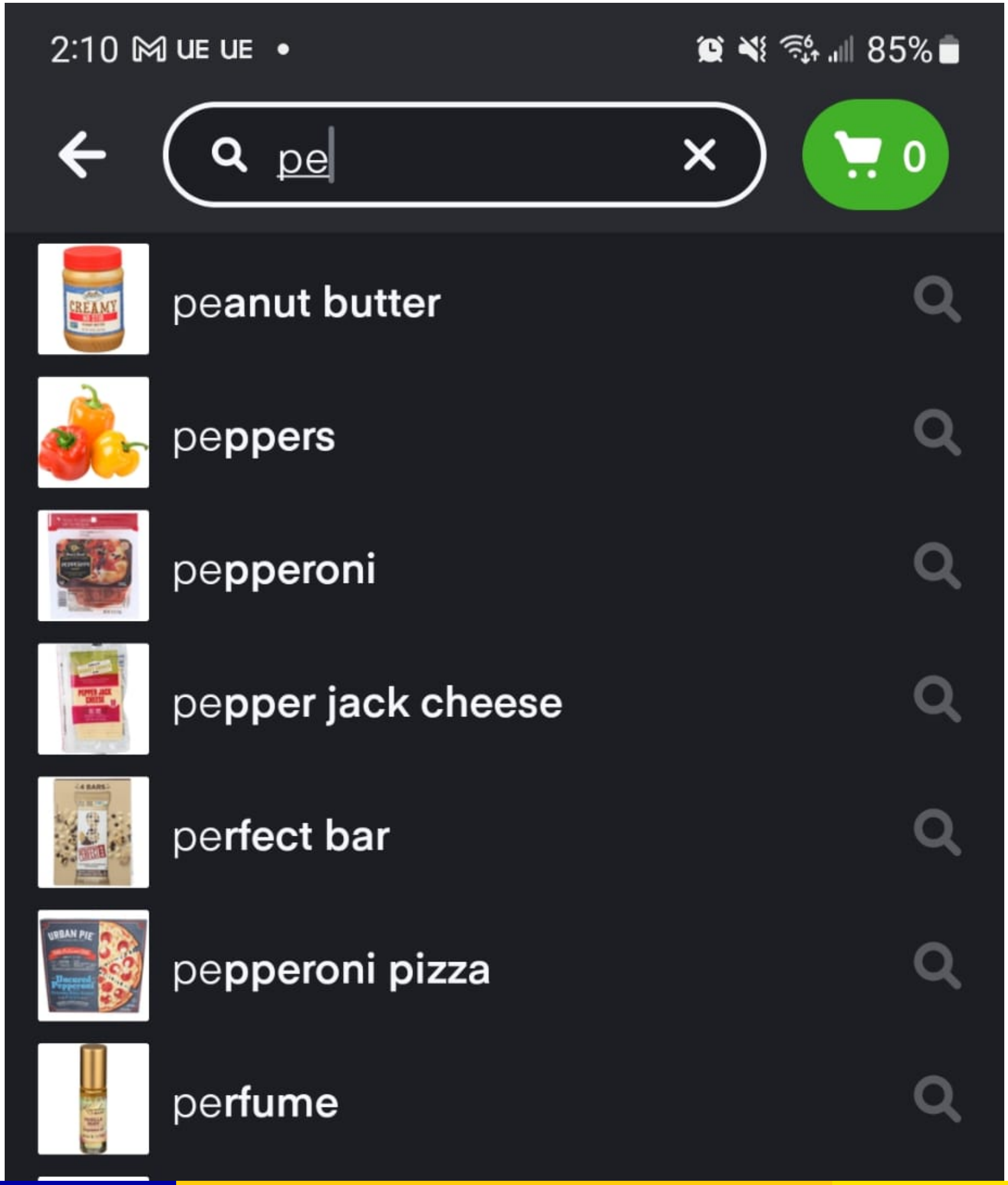


Figure 7. ITEMS system architecture.

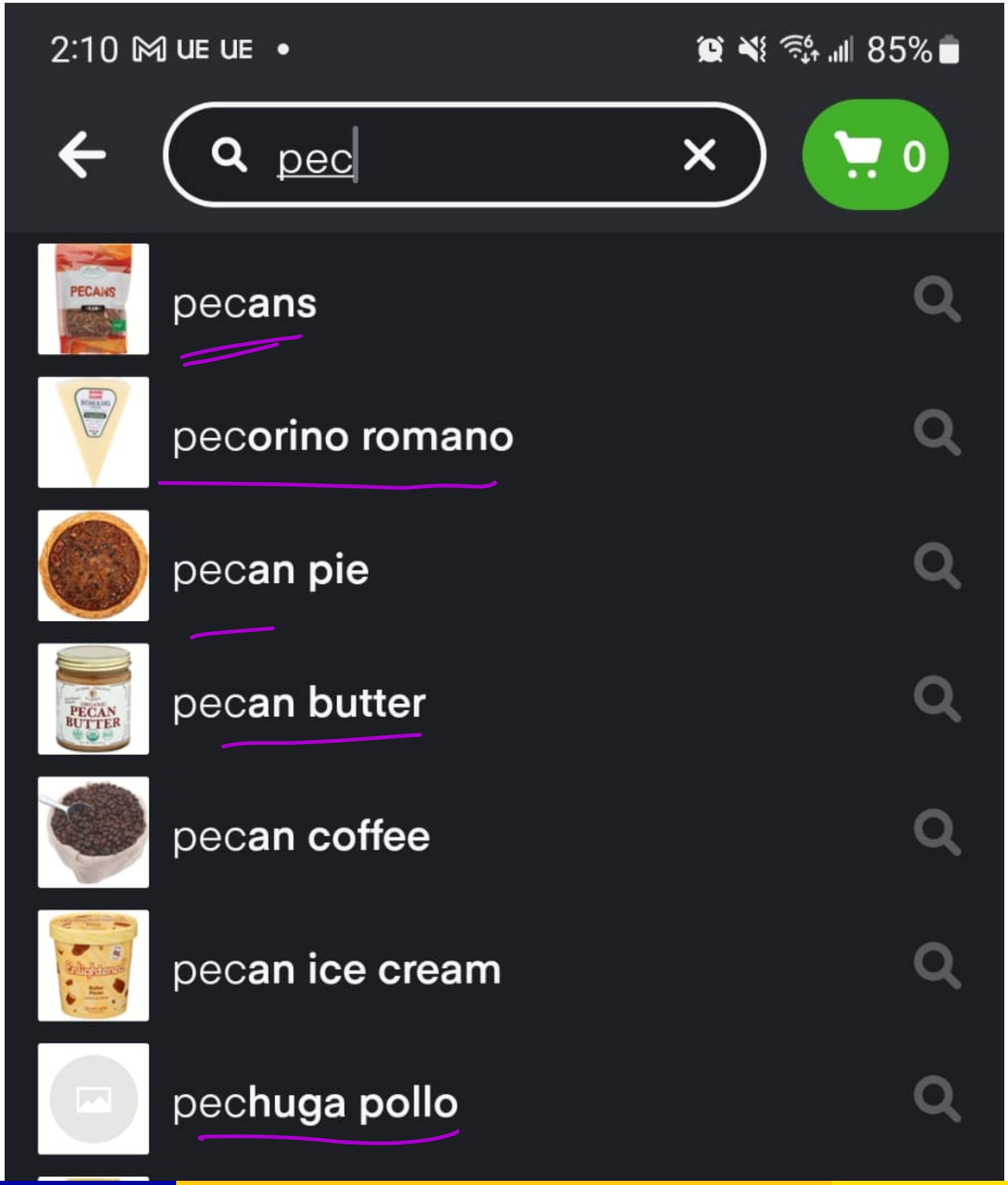
Instacart Auto-Complete



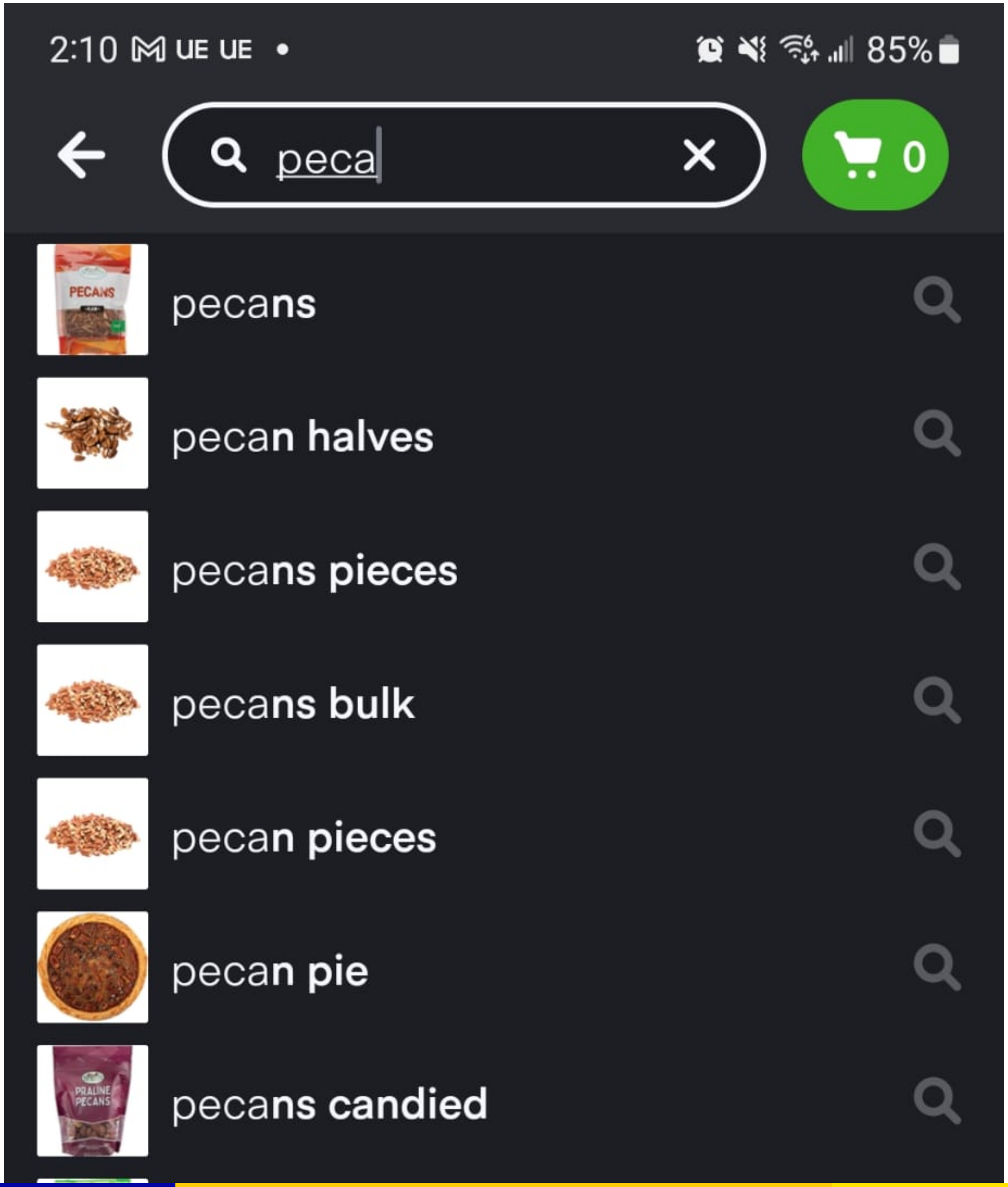
Instacart Auto-Complete



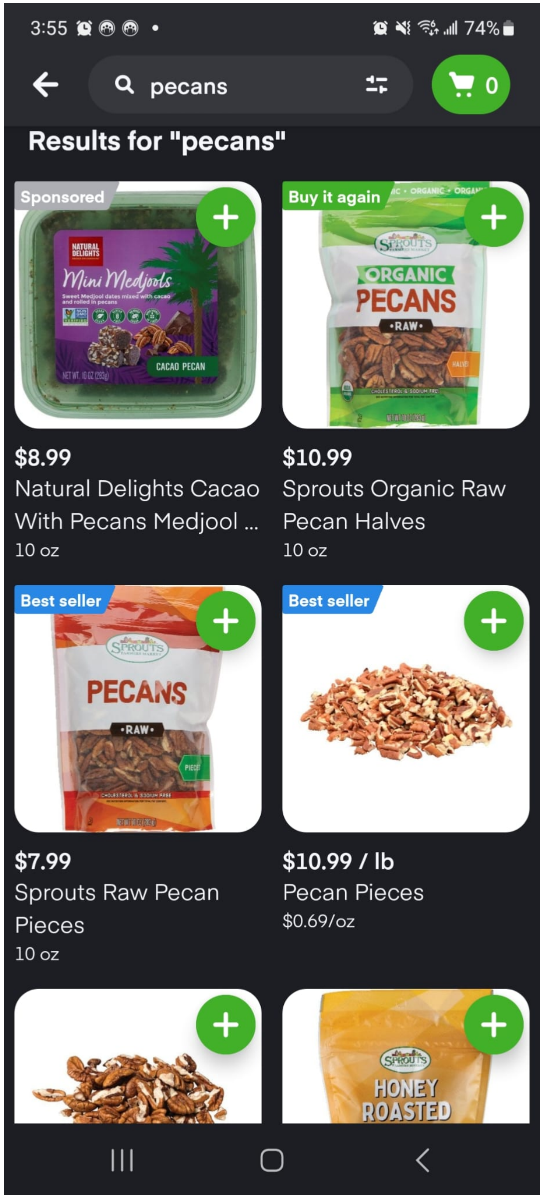
Instacart Auto-Complete



Instacart Auto-Complete



Instacart Auto-Complete and Search Results



Instacart Diversifying Auto-Complete

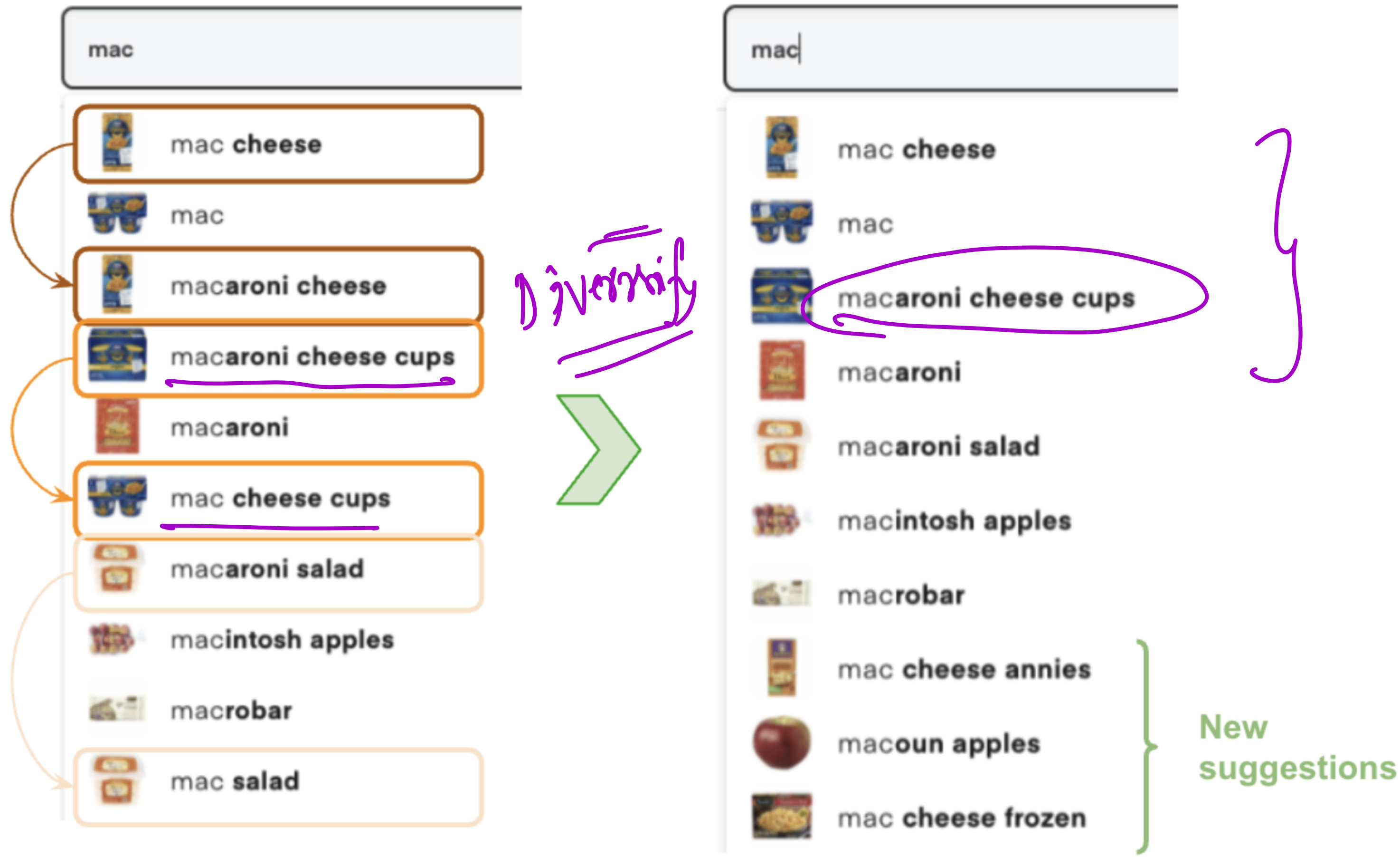


Figure 9. Autocomplete when a customer searches for "mac", before (left) and after (right) semantic deduplication.

Fine-Tuning BERT for Sentence Paraphrasing Demo

Demo Notebook available on course page