Chapter 10.

Introduction

In this chapter, we will discuss a variety of ways that we can create and fine-tune an embedding model to increase its representative and semantic power.

Embedding Model - Unstructured textual data by itself is often quite hard to process. They are not values we can directly process, visualize, and create actionable results from. We first have to convert this textual data to something that we can easily process: numeric representations. This process is often referred to as embedding the input to output usable vectors, namely embeddings, as shown in Figure 10-1.

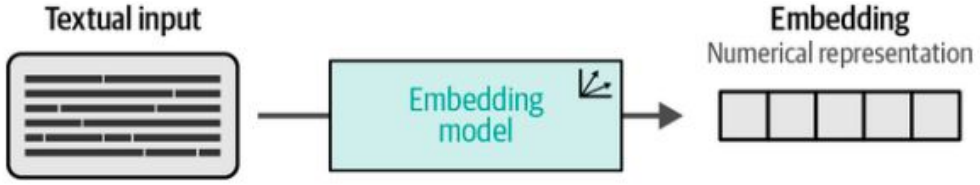

Figure 10-1. We use an embedding model to convert textual input, such as documents, sentences, and phrases, to numerical representations, called embeddings.

This process of embedding the input is typically performed by an LLM, which we refer to as an embedding model. The main purpose of such a model is to be as accurate as possible inrepresenting the textual data as an embedding.

Accuracy of an Embedding Model

However, what does it mean to be accurate in representation? Typically, we want to capture the semantic nature—the meaning—of documents. If we can capture the core of what the document communicates, we hope to have captured what the document is about. In practice, this means that we expect vectors of documents that are similar to one another to be similar, whereas the embeddings of documents that each discuss something entirely different should be dissimilar.

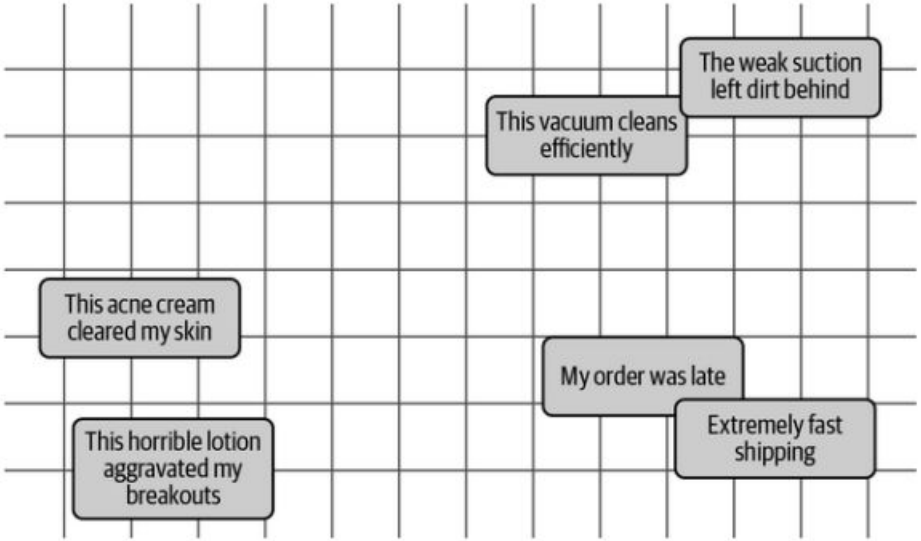

Figure 10-2. The idea of semantic similarity is that we expect textual data with similar meanings to be closer to each other in n-dimensional space (two dimensions are illustrated here).

Contrastive Learning

- One major technique for both training and fine-tuning text embedding models is called contrastive learning.
- Contrastive learning is a technique that aims to train an embedding model such that similar documents are closer in vector space while dissimilar documents are further apart.
- The underlying idea of contrastive learning is that the best way to learn and model similarity/dissimilarity between documents is by feeding a model examples of similar and dissimilar pairs.
- In order to accurately capture the semantic nature of a document, it often needs to be contrasted with another document for a model to learn what makes it different or similar.
-

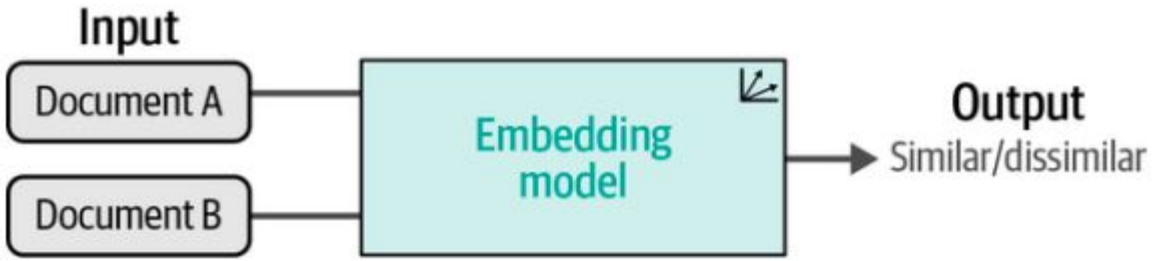

Figure 10-4. Contrastive learning aims to teach an embedding model whether documents are similar or dissimilar. It does so by presenting groups of documents to a model that are similar or dissimilar to a certain degree.

Contrastive Learning

- For example, you could teach a model to understand what a dog is by letting it find features such as “tail,” “nose,” “four legs,” etc. This learning process can be quite difficult since features are often not well-defined and can be interpreted in a number of ways. A being with a “tail,” “nose,” and “four legs” can also be a cat. To help the model steer toward what we are interested in, we essentially ask it, “Why is this a dog and not a cat?” By providing the contrast between two concepts, it starts to learn the features

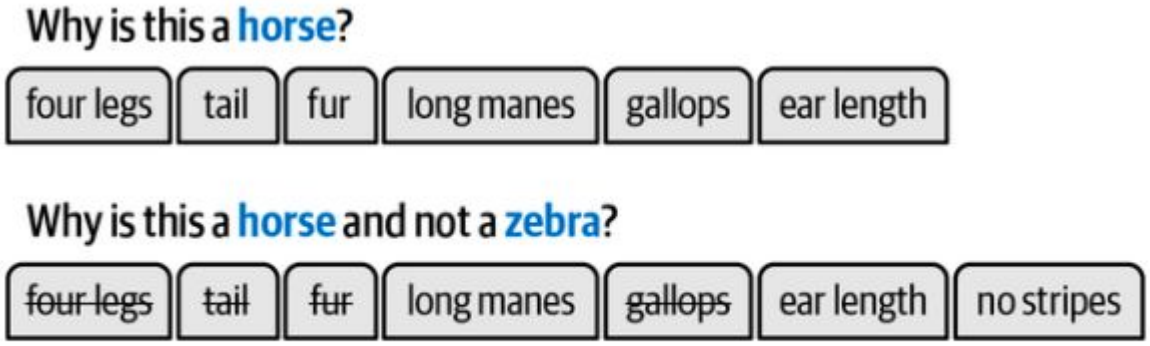

Figure 10-5. When we feed an embedding model different contrasts (degrees of similarity), it starts to learn what makes things different from one another and thereby the distinctive characteristics of concepts

SBERT

- Although there are many forms of contrastive learning, one framework that has popularized the technique within the natural language processing community is **sentence-transformers**.
- Before sentence-transformers, sentence embeddings often used an architectural structure called cross-encoders with BERT. A cross-encoder allows two sentences to be passed to the Transformer network simultaneously to predict the extent to which the two sentences are similar. It does so by adding a classification head to the original architecture that can output a similarity score.

- Unlike a cross-encoder, in sentence-transformers the classification head is dropped, and instead mean pooling is used on the final output layer to generate an embedding. This pooling layer averages the word embeddings and gives back a fixed dimensional output vector. This ensures a fixed-size embedding.

SBERT

The training for sentence-transformers uses a Siamese architecture. In this architecture, as visualized in Figure 10-7, we have two identical BERT models that share the same weights and neural architecture.

These models are fed the sentences from which embeddings are generated through the pooling of token embeddings. Then, models are optimized through the similarity of the sentence embeddings.

Since the weights are identical for both BERT models, we can use a single model and feed it the sentences one after the other.

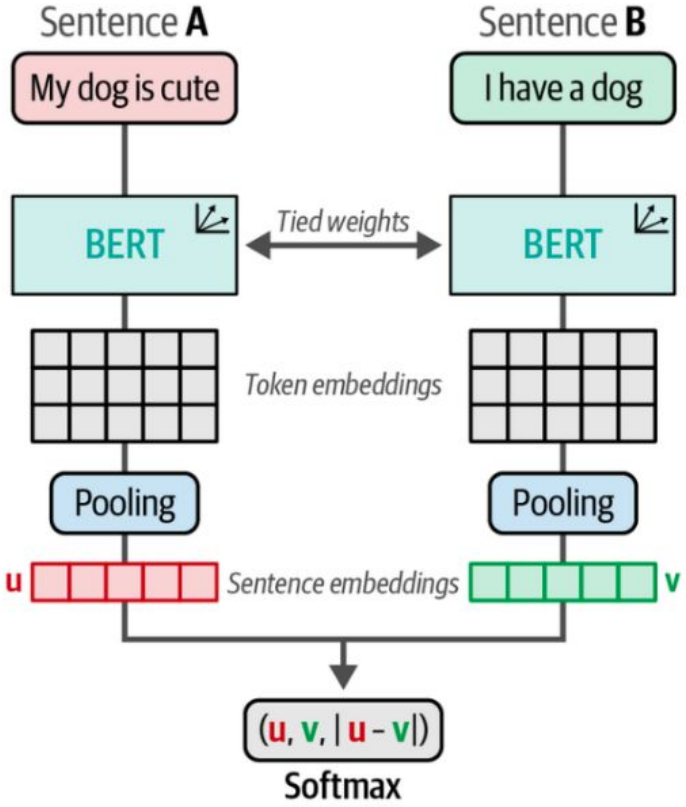

Figure 10-7. The architecture of the original `sentence-transformers` model, which leverages a Siamese network, also called a bi-encoder.

The resulting architecture is also referred to as a bi-encoder or SBERT for sentence-BERT. Although a bi-encoder is quite fast and creates accurate sentence representations,

Steps to create an embedding Model

Step - 1. Generating Contrastive Examples - When pretraining your embedding model, you will often see data being used from natural language inference (NLI) datasets. NLI refers to the task of investigating whether, for a given premise, it entails the hypothesis (entailment), contradicts it (contradiction), or neither (neutral).

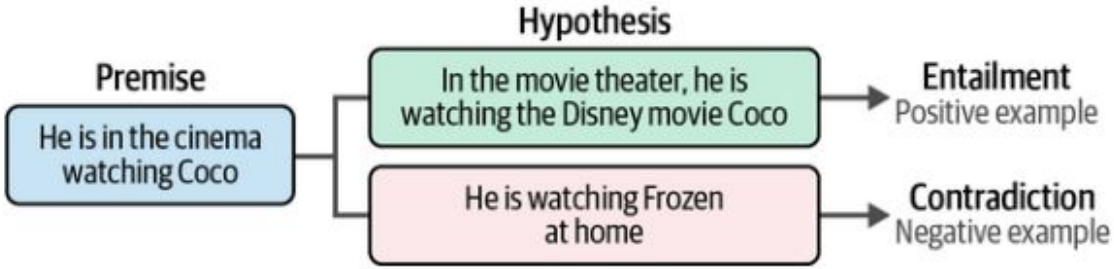

Figure 10-8. We can leverage the structure of NLI datasets to generate negative examples (contradiction) and positive examples (entailments) for contrastive learning.

Similar statements are Entailment and different statements are contradictions.

If you look closely at entailment and contradiction, then they describe the extent to which two inputs are similar to one another. As such, we can use NLI datasets to generate negative examples (contradictions) and positive examples (entailments) for contrastive learning.

Steps to create an embedding Model

Step - 2. Train Model - Now that we have our dataset with training examples, we will need to create our embedding model. We typically choose an existing sentence-transformers model and fine-tune that model,

Step - 3. In-Depth Evaluation of the Model - Benchmark (MTEB) was developed to compare embedding models. The MTEB spans 8 embedding tasks that cover 58 datasets and 112 languages. To publicly compare state-of-the-art embedding models, a leaderboard can be created with the scores of each embedding model across all tasks: