In [1]:
```
%run Latex_macros.ipynb
```

References

- Triplet Loss: [FaceNet: A Unified Embedding for Face Recognition and Clustering (https://arxiv.org/pdf/1503.03832.pdf)](https://arxiv.org/pdf/1503.03832.pdf)
- Sentence BERT: [entence-BERT: Sentence Embeddings using Siamese BERT-Networks (https://arxiv.org/pdf/1908.10084.pdf)](https://arxiv.org/pdf/1908.10084.pdf))

Embeddings

We speculate that a Neural Network is creating an alternate representation of the input

- into a latent space that enables a Head Layer (e.g., Classifier) to do its work
- training the model produces a representation that has features useful to the Head to complete its task (e.g., Classification)

We will refer to these alternate representations as *embeddings*.

When we plot embeddings in the latent space

- we might hope to see clustering of examples that are related

For example, here is a plot of a subset of the 10 digits in a 2D latent space

And here is the clustering of text articles across different classes.

If such clusters were associated with class labels, the Classifier Head's job would be facilitated.

</table>
Attribution: https://joeddav.github.io/blog/2020/05/29/ZSL.html (https://joeddav.github.io/blog/2020/05/29/ZSL.html)

We also hypothesize that

- that *intermediate* layers (distance greater than 1 from the Head) produ meaningful embeddings
- in Neural Style Transfer we hypothesized
    - the representation of shallow layers captures "syntax" (e.g. (
    - the representation of deeper layers captures "semantics" (e.

What does clustering enable ?

If a NN produced embeddings such that had the *desirable property* that

- the distance between the embeddings of related examples
- was *closer*
- than the distance between the embeddings of unrelated examples

what could we do ?

Zero-shot classification

Given an example and a set of possible labels

- using a pre-trained NN
- embed the example
- embed each of the labels

The label whose embedding was closest to that of the example would hope
correct label for the example.

This is *zero shot*

- since we are not fine-tuning
- or changing the weights
- of the pre-trained NN used to create the embeddings

A simple example: facial (or image) recognition

Semantic search

Want to create your own search engine ?

- create embeddings (using a NN for NLP) for each document
- create an embedding for your query

The document whose embedding is closest to the query's embedding woul
the correct result.

Note

This is the basis for *Vector Stores*

- augmenting a LLM with your own data (e.g., GPT)

Creating embeddings for similarity

The problem is that the hoped-for desirable property *may not be true* withc
requiring that in training or fine-tuning.

We can train a Neural Network to have this property by

- creating a Loss function to express this objective

One such objective is the [Triplet Loss (https://arxiv.org/pdf/1503.03832.p

Consider an input a (the "anchor")

- with related input p ("positive")
- with unrelated input n ("negative")

The Triplet Loss objective is to *minimize*

$$\max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$$

The loss is minimized when

- s_a is close to s_p
- s_a is far from s_n

That is the embedding for anchor

- a is very similar to that for p
- a is very dissimilar to that for n

Example: Sentence Embeddings

To illustrate, we show Sentence BERT (https://arxiv.org/pdf/1908.10084.p

- fine-tunes the embeddings produced by BERT
- in order to make related sentences close in embedding space

ϵ is called the *margin*

Sentence-BERT builds a network based on the embeddings produced by B

- how much farther the negative n must be from anchor a than positiv

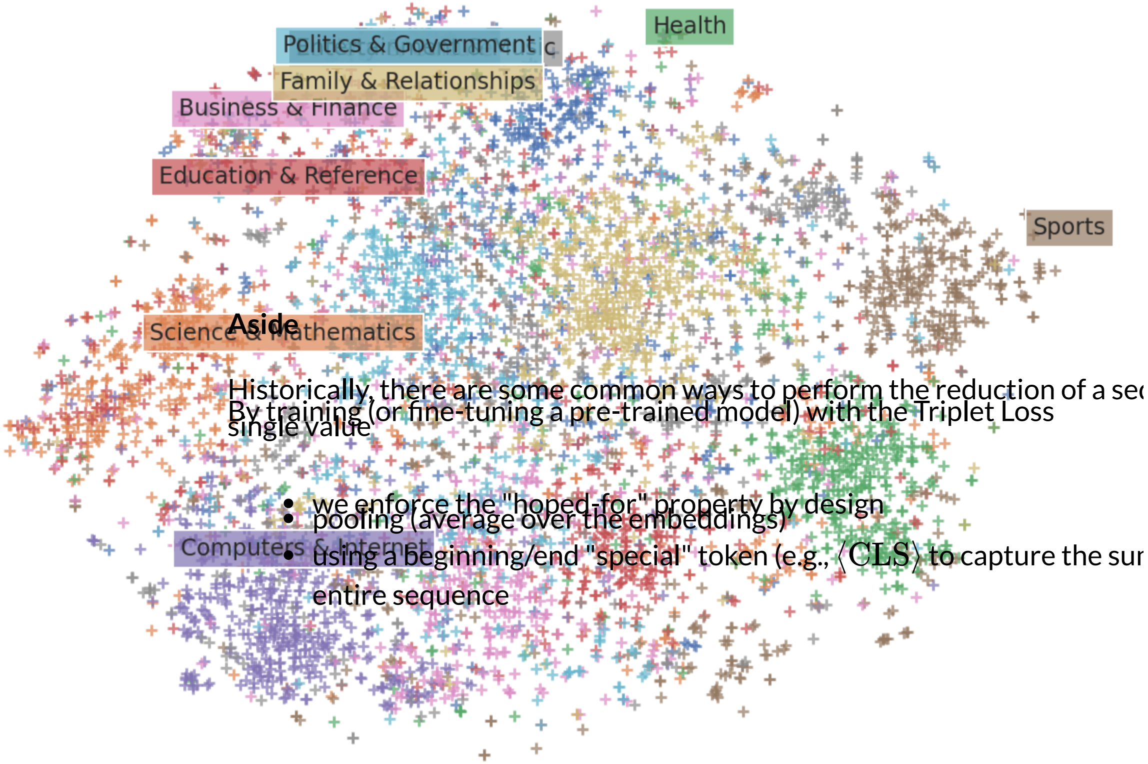
BERT is an Encoder style transformer.

A Transformer Encoder creates a Context Sensitive embedding of each inp

$$
\begin{array}{ll}
\text{if} & \max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0) < 0 \\
\text{then} & ||s_a - s_p|| - ||s_a - s_n|| + \epsilon < 0 \\
\text{re-arranging} & ||s_a - s_p|| < ||s_a - s_n|| - \epsilon
\end{array}
$$

so $||s_a - s_n||$ must be at least ϵ greater than $||s_a - s_p||$.

This reduces the need for the optimizer to make a and p *exactly* coincide (so $||s_a - s_p|| = 0$)



Aside

Historically, there are some common ways to perform the reduction of a sec
single value

- pooling (average over the embeddings)
- using a beginning/end "special" token (e.g., $\langle \text{CLS} \rangle$) to capture the sur
entire sequence

By training (or fine-tuning a pre-trained model) with the Triplet Loss

- we enforce the "hoped-for" property by design

</table>
Attribution: [https://arxiv.org/pdf/1908.10084.pdf#page=3 (https://arxiv.org/pdf/1908.10084.pdf#page=3)](https://arxiv.org/pdf/1908.10084.pdf#page=3)

The pre-trained BERT model is *shared* across two inputs: Sentence A and Ser

- "weights are tied"
- BERT's weights are fine-tuned via the Triplet Loss objective

The sequence output of BERT is reduce by pooling (in this case)

- Sentence A is embedded as u
- Sentence B is embedded as v

In the diagram on the right, the Triplet Objective

- is recast as maximizing similarity (cosine distance)

Here is the architecture

Aside

The diagram on the left is for producing embeddings for a specific task

- entailment
    - Does Sentence B logically follow from Sentence A
- and hence is expressed as a Classification objective over labels $\{\text{"Entail"}, \text{"Does not entail"}\}$

The inputs to the classifier are the concatenation of

- the embedding u of Sentence A
- the embedding v of Sentence B
- the difference in the embeddings

(Presumably these three inputs facilitate Classsification)