Tomas Anthony CS 410

Tech Review - SentenceTransformers

In 2018 Google released Bidirectional Encoder Representations from Transformers (BERT) to the world. The technique revolutionized natural language processing and became the foundation of a large portion of new NLP experiments (Rogers et al., 2020, pg 854). Some of these experiments focused on improving BERT by focusing on areas of weakness. One of the largest weaknesses of BERT is its resource intensive nature. The base version of BERT contains 110 million model parameters (Sun et al., 2019, pg 1) and is too "computation-intensive to suit low-capability devices or applications with strict latency requirements" (Ganesh et al., 2021, pg 1).

In order to address this weakness, researchers began investigating methods to compress the BERT model or to reduce inference time. One of these efforts produced Sentence-BERT (SBERT), a BERT derived model created specifically to enable the practical use of BERT for "large-scale semantic similarity comparison, clustering, and information retrieval via semantic search" (Reimers & Gurevych, 2019, pg 1). SBERT produces sentence embeddings for text, semantically meaningful vectorizations of words, that are then comparable using simple cosine similarities (Reimers & Gurevych, 2019, pg 1). Following the release of the SBERT paper, its authors released the code behind the paper in a library named SentenceTransformers. SentenceTransformers is an easy to use toolkit to implement state of the art, production friendly NLP processing.

SentenceTransformers is a Python framework that is used in tandem with variations of the BERT model that produce useful word embeddings. The documentation on the SentenceTransformers website, sbert.net, is extensive (*SentenceTransformers documentation,* 2019) and provides thorough tutorials. The documentation provides installation, quickstart, usage, and training guides. The SentenceTransformers team also supplies a repository of pretrained models for use with their framework (*Pretrained models,* 2019). The SentenceTransformer does an excellent job of providing everything needed to start working with SBERT right away.

To load a model with the sentence_transformers package, a user simply needs to initialize the SentenceTransformer class with the name or path of the model to be used.

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-mpnet-base-v2')
```

If the parameter passed to the class constructor is a model name and not a path, then sentence_transformers will automatically download the model at execution time. The model name must be a valid sentence_transformers model.

Once the SentenceTransformer object is created, there is one main function that can be used to leverage the SBERT style sentence embeddings - the encode function. An array of texts- documents, single words, sentences, etc - is passed into the encode function and then returns an array of the vectorized sentence embeddings of the same text.

```
to_be_sberted = ['CS 410 is the best UIUC course',
    'Text Information Retrieval is cool',
    'Writing papers is so much fun.’]

other_sbertable = ['Another sentence.',
    'Let's compare this sentence as well',
    'Sure why not, another one.’]

first_encodings = model.encode(to_be_sberted)
second_encodings = model.encode(other_sbertable)
```

The sentence embeddings can then be used in conjunction with the scikit-learn library’s cosine_similarity function or other analogous functions to compare semantic similarities or conduct other defined usages of SBERT..

```
from sklearn.metrics.pairwise import cosine_simiilarity

similarities = cosine_similarity(first_encodings,
second_encodings)
```

Returning the max values in the cosine similarity matrix would then provide the most semantically similar pairings for each sentence. Returning the top three would give a ranked result. These values can be used in different ways, but I have primarily used it for quick, bootstrapped solutions for semantic textual similarity, semantic search, and even named entity recognition.

For my purposes and my experience with the SentenceTransformer framework and SBERT, I have found the library easy to use and highly effective. It takes minimal code to get a working solution for a given problem, is easily modifiable to use a different pretrained model, and is highly scalable for deploying into a production environment. SBERT and SentenceTransformers is an excellent adaptation of BERT techniques for

NLP use cases and addresses perhaps BERT's biggest weakness - its computational usage and size.

I would highly recommend the SentenceTransformers library for anyone interested in implementing state of the art natural language processing solutions, specifically, usages that lend themselves to the effective leverage of generated text embeddings.

Citations

Ganesh, P., Chen, Y., Lou, X., Khan, M. A., Yang, Y., Sajjad, H., Nakov, P., Chen, D., & Winslett, M. (2021). Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, *9*, 1061–1080. https://doi.org/10.1162/tacl_a_00413

Pretrained models. Pretrained Models - Sentence-Transformers documentation. (n.d.). Retrieved November 8, 2021, from https://www.sbert.net/docs/pretrained_models.html.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using Siamese Bert-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. https://doi.org/10.18653/v1/d19-1410

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in Bertology: What we know about how Bert Works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866. https://doi.org/10.1162/tacl_a_00349

SentenceTransformers documentation. SentenceTransformers Documentation - Sentence-Transformers documentation. (n.d.). Retrieved November 8, 2021, from https://www.sbert.net/index.html.

Sun, S., Cheng, Y., Gan, Z., & Liu, J. (2019). Patient knowledge distillation for Bert Model Compression. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. https://doi.org/10.18653/v1/d19-1441