Photo by Jessica Ruscello on Unsplash

This article covers sentence embeddings and how codequestion built a fastText + BM25 embeddings search. Source code can be found on github.

Natural language processing (NLP) is one of the fastest growing areas in the field of machine learning. Deep innovation is happening on many fronts, leading to users being able to find better data faster. Techniques and algorithms once only realistic for those with massive IT budgets and resources are now able to run on a laptop. NLP is not a new problem though and there are time-tested methods available that perform very well. The simplest solution many times can be the best solution.

This article will explore various methods and go through an evaluation process for building a text search system.

Full-text search engines

Full-text search engines that allow users to enter a search query and find matching results are a reliable solution with a great performance history. In these systems, each document is tokenized usually with a list of common words removed referred to as stop words. Those tokens are stored in an inverted index and each token is weighed based on metrics like token frequency.

Search queries are also tokenized using the same method and the search engine finds the best matching record for the query tokens. There are highly-distributed and very fast solutions that have proved themselves in production for many years, like Elasticsearch. The most common token weighting algorithm today is BM25. It works very well and is still hard to beat.

Embeddings

Word embeddings have rapidly evolved over the last 5–7 years, first starting with Word2vec, followed by GloVe and fastText. These algorithms all typically build 300 dimension vectors. More advanced embeddings are now available that are larger in size (768+ dimensions) and able to understand the context of words within a sentence, with BERT embeddings leading the way. In other words, advancing past simple bag of word methods.

Easy to use, robust libraries are out there that enable developers to take advantage of these advancements. HuggingFace Transformers is an excellent library with a fast-growing set of cutting-edge functionality. Universal sentence embeddings is another important area of development. Having a single dynamic model that can transform text into embeddings for downstream learning tasks is crucial. Sentence Transformers is built on top of Transformers and is an excellent example of a library that can build high-quality sentence embeddings.

Performance considerations

As the march to build the perfect embeddings races forward, what does that mean for practitioners trying to solve current business problems. Do you always use the latest and greatest advancements? Do you always needs to? There are trade offs with more complexity. When processing millions and billions of data points, processing time vs acceptable functionality is always a conversation that needs to happen. These conversations factor in resources available, time available and what is necessary to satisfy the requirements for a given problem.

Some questions to ask when evaluating a text search solution.

- How fast do I need query responses and indexing to be? How high of accuracy is required?
- What compute resources do I have access to? Do I have access to GPUs?
- How many records vs how much storage do I have? Assuming vectors are using 32 bit (4 byte) floats, a 300 dimensional vector takes 300x4 bytes per record. For a 4096 dimensional vector, it's 4096x4, which requires 13 times more space for the same data.
- Can a full-text search engine satisfy the requirements?

What are we building?

codequestion is an application that allows a user to ask coding questions directly from the terminal. Many developers will have a web browser window open while they develop and run web searches as questions arise. codequestion works to make that process faster so you can focus on development. It also allows users working without direct internet access or reduced internet access the ability to run code question queries. An example of the application in action is below.