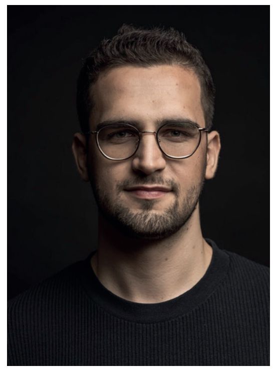
ROSENHEIMER INFORMATIKPREIS
INF-MASTER



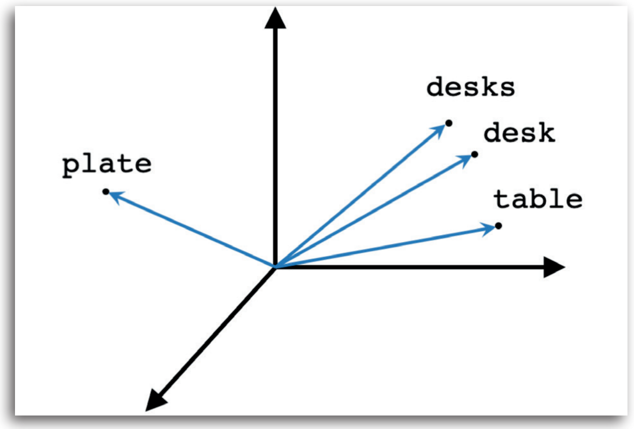
COMPRESSION OF SEMANTIC VECTORS INTO BIT VECTORS TO ENABLE EFFICIENT SEARCH WITH LARGE LANGUAGE MODELS

YANNIK HINTEREGGER (MASTERSTUDIUM INFORMATIK)
Betreuer: Prof. Dr. Markus Breunig, Prof. Dr. Marcel Tilly

The term Large Language Model (LLM) is constantly mentioned when discussing AI. Today, w the use of LLMs, writing emails, code, or even an entire bachelor's or master's thesis is now easy. However, many there are many other aspects to these language models that are less prominently discussed. For example, language models can be used to build highly effective search engines.

Comparing texts is not simple – especially for machines. Consider these sentences: “The weather is nice” and “The sun smiles at me today.” Although the meanings are the same, they do not share any words. From context, we humans can recognize the similarity, but a machine struggles with this.

With language models it is possible to encode the meaning of language into vectors – known as embeddings.
These embeddings encode the meaning of a text using the direction of a vector. While this may sound abstract, the fundamental idea is straightforward: The vectors of similar sentences point into similar directions.



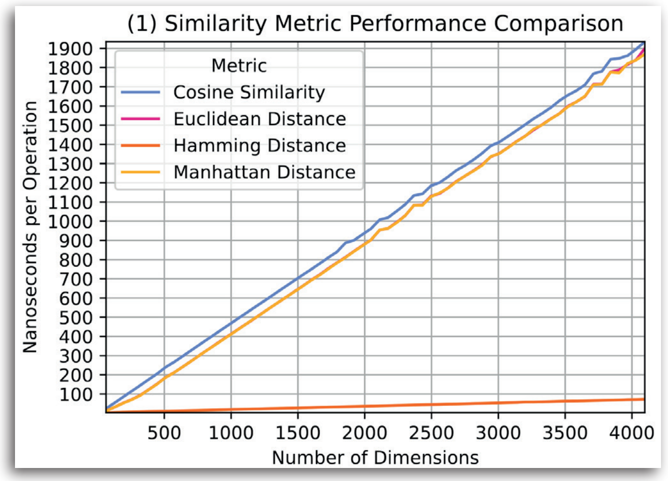
Four words represented by their embeddings in a continuous 3-dimensional vector space.[1]

It is now widely accepted that LLMs are very good at encoding the meaning of language into these embeddings.
Utilizing these embeddings has significantly improved various language processing tasks. However, these embeddings face a challenge: scalability.

The embeddings of LLMs are high-dimensional, often having between 500 and 2000 dimensions in a real-valued space. For example, the well-known language model BERT has embeddings consisting of 768 floats. This means an embedding requires slightly more than 3 KB of memory. At first glance, this doesn't seem like much, but large text databases quickly accumulate immense data volumes. Consider the English Wikipedia, which consists of 210 million sentences. Making each sentence searchable would result in 650 GB of vectors.

Another issue is processing time. To determine the similarity between two vectors, cosine similarity is used, which essentially measures the angle between two vectors. Using a Java implementation of cosine similarity, it takes about 360 nanoseconds to compare two vectors. Applying this to the Wikipedia example, without further optimizations such as a data structure like HNSW, it would take approximately 75 seconds of CPU time to determine the similarity of one vector to all others.

It's important to note that this processing speed is only achievable if all embeddings are stored in memory.
Therefore, it requires a machine with a lot of memory and computing power to handle a text database the size of the English Wikipedia. Even then, response times of 75 seconds are poor for a user, and multiple users cannot be processed simultaneously.

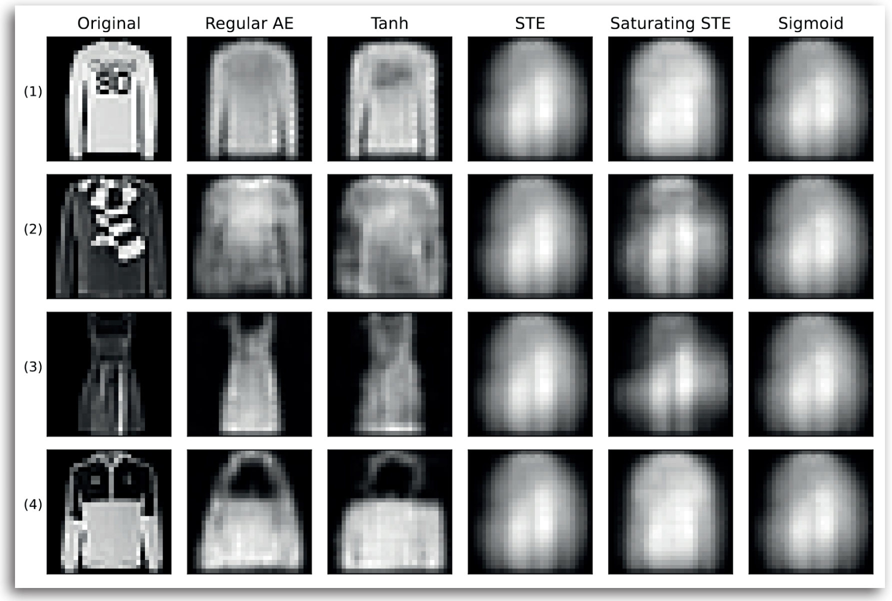


[1] Pilehvar, M. T., & Camacho-Collados, J. (2020). *Embeddings in natural language processing: Theory and advances in vector representations of meaning.* Morgan & Claypool Publishers

This problem motivated my master's thesis. In my thesis, I explored methods to reduce the memory requirements and processing time of the float embeddings generated by LLMs.

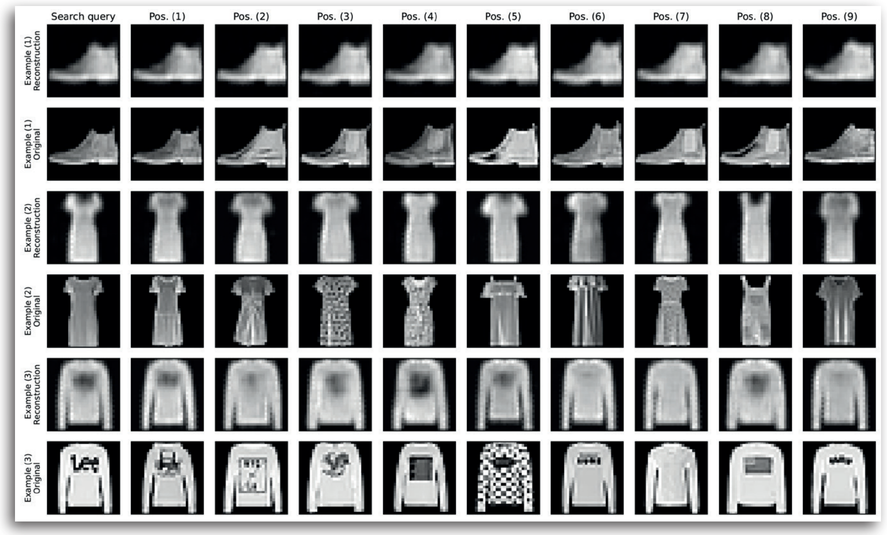
We first examined other similarity measures for vectors, implemented them in Java and compared their performance. As shown in the figure below, Hamming distance is significantly faster. This speed is due to comparing binary vectors using an XOR operation, which CPUs can execute very efficiently since XOR is a basic CPU operation, physically implemented by the transistors on a CPU.

However, the embeddings of LLMs are represented in a real space, not a binary space. Therefore, we investigated how binary autoencoders – a specific type of AI algorithm – can be used to transfer embeddings from a real space to a binary space.



In my master's thesis, we demonstrated that binary autoencoders could achieve a very high compression rate for embeddings if a slight semantic quality loss is acceptable. For example, BERT's 768-dimension float embeddings were translated into 512-dimension binary embeddings. This translation resulted in a median quality loss of just over 3%, a 46-fold reduction in memory requirements, and a 30-fold reduction in processing time. This means a search query against the English Wikipedia could be handled with 13.5 GB of memory and 2.4 seconds of CPU time.

Furthermore, we showed that binary autoencoders could generate binary embeddings for other data types, such as images. We implemented an image similarity search using binary embeddings. In the subsequent figures, the original images are in the first column on the left, and the encoded information of the images is shown to the right. The right figure shows a search implemented with these binary embeddings. The left image is the query image, and the images to the right are sorted by similarity of their binary embeddings.



BESTE ABSCHLUSS- ARBEITEN 2023/24