ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/priyank-anurag/120

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science
https://urfpublishers.com/journal/artificial-intelligence

Vol: 2 & Iss: 1

Review Article

Review of Word Embeddings, their Classification, and use Case in LLM Application

Priyank Rathod* and Anurag

Intel Corporation, Folsom, CA, USA

Citation: Rathod P, Anurag. Review of Word Embeddings, their Classification, and use Case in LLM Application. *J Artif Intell Mach Learn & Data Sci 2024*, 2(1), 431-433. DOI: doi.org/10.51219/JAIMLD/priyank-anurag/120

Received: 01 March, 2024; **Accepted:** 18 March, 2024; **Published:** 20 March, 2024

***Corresponding author:** Priyank Jayantilal Rathod, Intel Corporation ,Folsom,CA, USA, E-mail: rathodpriyank@gmail.com

ABSTRACT

Large Language Models (LLMs) have revolutionized the field of natural language processing, enabling AI systems to understand and generate human-like language. A crucial component of LLMs is word embeddings, which allow them to capture the semantic meaning of words, sentences, and documents in high-dimensional vectors. These models have made significant impacts on various NLP tasks like sequence labeling and text classification. This report compares various vector embeddings, their importance in LLMs, and benchmarks used in the scientific community to evaluate their performance.

Keywords: Large Language Models (LLM), Word Embeddings, NLP, Vector Database

1. Introduction

Word embeddings are the building blocks of LLMs, enabling models to capture semantic relationships between words, understand context, and improve performance on various natural language processing tasks. By representing words as n-dimensional vectors, embeddings provide a semantic representation of words, capturing contextual nuances, which helps to understand language semantics, improve text generation, and enhance performance on downstream tasks. In technical terms, a word embedding is defined as a function that maps each word (w) from a vocabulary (V) to a real-valued vector () in a D-dimensional embedding space. The similarity between two-word vectors is commonly measured using cosine similarity. The list of nearest neighbors for a word (w) includes all other words (v) from the vocabulary(V), except for (w) itself. These words are arranged from most similar to least similar to (w) based on how closely they are related or similar in meaning[1,2]. The Transformer model, a popular LLM architecture, relies heavily on embeddings to encode input tokens and positions within a sequence. This enables the model to learn complex patterns and relationships in the data. The tokenization process breaks down input sentences into tokens, which are then transformed into vectors, known as vector embeddings. Choosing the right embedding model is crucial for LLM applications. Affordable alternatives to OpenAI's API, such as Langchain, optimize LLM applications with vector embeddings. This enables faster and more efficient representation learning, improving performance on downstream tasks such as Text Classification, Named Entity Recognition, Question Answering, Text Generation, etc. The availability of myriad embedding models makes it crucial to choose the most effective embedding model to obtain better performance in any Natural Language Processing (NLP) applications.

2. Classification of Word Embedding Models

Word embeddings can be broadly categorized into static word embeddings and Contextualized word embeddings[3] as explained further below:

Static Word Embedding: static embedding techniques encode words as fixed dense vectors. These embeddings are "static" because each word has one representation that does not change, meaning the model cannot capture the word's polysemy. These models learn a single embedding for each word in the vocabulary, regardless of the context in which the word is used. Such as:-

- Word2Vec, one of the most widely used word embedding models developed by[4]. It uses a two-layer neural network to learn word associations from a large text corpus and represents these words in a vector space.
- **Glove:** GloVe is another popular word embedding model developed by[5]. It uses a matrix factorization technique to learn word embeddings.
- **FastText:** FastText is a scalable word embedding model developed by[6]. It uses a convolutional neural network to learn word embeddings.
- **Contextualized word embedding:** contextualized embedding generates dynamic representations for words by considering their contextual information within a sentence. Embeddings are "contextualized" because the representation of a word can change depending on its usage in different sentences, allowing the model to effectively capture the nuances of word meanings and polysemy. Some popular models are:-
- **Elmo:** Elmo is a contextualized word embedding model developed by[7]. It uses a multi-task deep neural network to learn word embeddings.
- **GPT-2:** GPT-2 is a large-scale language model developed by[8]. It uses a transformer architecture to learn word embeddings.
- **BERT:** BERT (Bidirectional Encoder Representations from Transformers) is a contextualized word embedding model developed by[9]. It is a powerful language model that uses a multi-layer bidirectional transformer encoder and a multi-task learning approach to generate word embeddings.

The choice between static and contextualized word embeddings should be guided by the corpus size and the analysis task's specific needs. Contextualized embeddings offer advantages in capturing context-sensitive meanings and are preferable for smaller corpora or tasks requiring a nuanced understanding of language, while static embeddings trained only on the focus corpus capture opposing opinions better than contextualized embedding[3]; hence, they might be practical for large-scale analyses where the focus is on capturing related concepts or when computational resources are a concern. Different embeddings offer varying speeds and context awareness, impacting the model's overall performance.

3. Benchmark for Evaluating Word Embeddings

Evaluating the performance of word embeddings is essential to understanding their effectiveness in capturing the properties of words. Two main approaches are used in evaluating word embeddings, namely, intrinsic evaluation and extrinsic evaluation. Intrinsic evaluation evaluates word embeddings based on their inherent properties and linguistic characteristics. It does not involve using the embeddings in real-world applications but instead tests them on specific linguistic tasks, whereas Extrinsic Evaluation involves assessing word embeddings in the context of real-world NLP tasks to determine their practical utility and effectiveness. The embeddings are evaluated based on their impact on the performance of these tasks[10]. This paper focuses on analyzing performance based on the Intrinsic Evaluation of embedding models. Intrinsic Evaluation can include evaluation based on Semantic Similarity, Analogical Reasoning, and Word Similarity. The most extensive benchmark under intrinsic evaluation is the Massive Text Embedding Benchmark (MTEB), a comprehensive framework designed to evaluate the performance of text embedding methods across a wide range of tasks and languages.

The MTEB encompasses eight embedding tasks, covering 58 datasets and 112 languages, making it the most extensive benchmark to date[11]. Its key components, like diverse tasks and datasets, comprehensive language coverage, open-source, and extensible and holistic evaluation, make it a prominent choice for benchmarking embedding models. MTEB offers a standardized framework for evaluating text embeddings, facilitating direct comparisons between models and approaches. Its wide range of tasks and language coverage helps us understand how well different embedding methods generalize across various NLP applications and linguistic contexts. The benchmark's comprehensive evaluation provides valuable insights into the strengths and weaknesses of different embedding models, helping select the most appropriate model for their specific needs. This encourages further research and development in text embeddings by identifying gaps and areas where no model performs optimally. As per the research conducted in the paper[11], the best-performing models varied based on the specific criteria considered. The glove was noted for offering maximum speed and performance, whereas models like GTR-XXL, ST5-XXL, or SGPT-5.8B were highlighted for their high performance but slower speed. Fine-tuned models like MPNet and MiniLM led the middle cluster in terms of balancing speed and performance. Larger models with billions of parameters tend to dominate many MTEB tasks, but they come at a significant computational cost. While self-supervised large language models demonstrated promise in closing the performance gap in natural language generation tasks, they frequently required supervised fine-tuning to reach competitive embedding results. Different models excelled in specific tasks based on their design and capabilities. For example, models like ST5-XXL and OpenAI Ada Similarity demonstrated high performance in classification tasks, showcasing the importance of selecting models tailored to the task. **(Table 1)** lists some of the best text embedding models, as of the writing of this paper, based on the MTEB benchmark taken from the leaderboard hosted on huggingface.co website[12].

Although MTEB offers a comprehensive perspective on the performance of text embeddings, it's also important to look at the individual scores on the tasks and datasets that best represent one's use case, highlighting the need for task-specific model selection and the ongoing evolution of text embedding methods.

Table 1. MTEB benchmark leaderboard.

Rank	Model	Average Score (56 Datasets)
1	voyage-large-2-instruct	68.28
2	SFR-Embedding-Mistral	67.56
3	gte-Qwen1.5-7B-instruct	67.34
4	voyage-lite-02-instruct	67.13
5	GritLM-7B	66.76

6	e5-mistral-7b-instruct	66.63
7	google-gecko.text-embedding-preview-0409	66.31
8	GritLM-8x7B	65.66
9	gte-large-en-v1.5	65.39
10	LLM2Vec-Mistral-supervised	64.8

4. Factors in Selecting Word Embedding Model

Selecting the correct text embedding model is critical in NLP applications. When choosing a text embedding model, it is essential to consider the task at hand. Different tasks, such as semantic textual similarity calculation, require distinct embedding models that cater to their specific needs. Leveraging MTEB to evaluate models across multiple languages and varied tasks could be a starting point. Analyze how well the model can handle extensive knowledge bases and consider the model's capacity to differentiate between words with comparable semantic properties and nearby subjects. Some large models offer high performance but come with increased costs. In contrast, smaller models can balance power and cost efficiency. Consider the trade-off between latency and storage and prioritize accuracy, efficiency, and versatility when selecting a model. Transformer-based models fine-tuned for specific tasks or domains are likely more effective choices.

Power, cost-efficiency, performance impact, versatility, and real-world scenario applicability will help developers and researchers make informed project decisions. NLP applications can achieve better results by evaluating models using the MTEB and prioritizing accuracy, efficiency, and versatility.

5. Embeddings as Databases in LLMs

Word embeddings or vector embeddings are at the core of vector databases, which play a vital role in LLMs, enabling fast and efficient similarity searches in high-dimensional spaces, which is done by storing embeddings in an index to allow the database to run searches more rapidly. This is particularly important in applications that require real-time user interaction, such as chatbots or virtual assistants. Vector databases ensure that response generation, which depends on fetching relevant context or information represented as vectors, is quick and efficient.

Additionally, vector databases aid in translation memory, storing previous translations as vectors in a database, facilitating faster and more accurate machine translation.

6. Impact of Word Embedding Models on NLP Applications

Word embedding models have significantly improved the performance of text classification tasks such as sentiment analysis and spam detection. They have made it possible to develop powerful language models such as RNNs, LSTMs, and GPT-3. Word embedding models find applications in information retrieval, knowledge bases, chatbots, text generation, and text-to-data. By identifying key ideas and generating summaries, word embeddings have enabled algorithms that accurately reflect the content of a document. Word embeddings are used in speech recognition systems to improve the accuracy of speech-to-text models and in several search engines to improve the relevance of search results. These models are also applied to sequence labeling tasks like named entity identification and part-of-speech tagging. Translating text from one language to another is another area where word embedding models are utilized. By capturing the subtleties of language, word embedding models have enhanced sentiment analysis and made it feasible for applications like opinion mining and product review analysis.

7. Conclusion

Word embeddings are pivotal in improving the capabilities of Large Language Models (LLMs) for Natural Language Processing (NLP) tasks, offering a spectrum from static to contextualized embeddings. Static embeddings provide efficiency for large-scale applications, while contextualized embeddings offer nuanced language understanding for context-sensitive tasks. The Massive Text Embedding Benchmark (MTEB) is a crucial tool for evaluating and comparing the performance of various embedding models across functions and languages, emphasizing the importance of model selection tailored to specific needs. Additionally, embedding embeddings into vector databases significantly boosts the efficiency of real-time applications like chatbots and virtual assistants. Ultimately, the choice of word embeddings plays a critical role in leveraging the full potential of NLP technologies, marking a continuous evolution towards more sophisticated language understanding and processing.

8. References

1. Schnabel T, Labutov I, Mimno D, Joachims T. Evaluation methods for unsupervised word embeddings. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing ACL Anthol 2015.
2. Tomas M, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. arXiv 2013.
3. Sarakul W, Rutherford AT. Contextualized vs. Static Word Embeddings for Word-based Analysis of Opposing Opinions. 2023 20th International JCSSE 2023; 95-100.
4. Mikolov T, Correira C. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the 26th International Conference on Machine Learning 2013.
5. Pennington J, Socher R, Manning C. GloVe: Global Vectors for Word Representation. In Conference of the North American Chapter of the Association for Computational Linguistics 2014.
6. Jin R, Zhang J. FastText: A scalable and high-speed learning algorithm for classification and regression. In Proceedings of the 30th International Conference on Machine Learning 2016.
7. Peters ME, Neumann M, Iyyer M, et al. 2018. Deep Contextualized Word Representations. ACL Anthol, 2018.
8. Radford A, Language Models with a Hidden State (Language Models). Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics 2019
9. Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. arXiv 2019.
10. Shi Y, Zheng Y, Guo K, Zhu L, Qu Y. Intrinsic or Extrinsic Evaluation: An overview of word embedding evaluation. 2018 IEEE ICDMW 2018: 1255-1262.
11. Muennighoff N, Tazi N, Magne L, Reimers N. MTEB: Massive Text Embedding Benchmark. arXiv 2023; 2014-2037.
12. https://huggingface.co/spaces/mteb/leaderboard.