International Journal of Informatics and Data Science Research
ISSN 2997-3961 (Online)
Vol. 2, No. 6, Jun 2025,
Available online at: https://scientificbulletin.com/index.php/IJIDSR

Optimization of Chatbot Intent Classification: a Comparative Analysis of NLP and LLM Embedding Techniques

Anvar Zokhidov
Former Senior AI Manager, Tenge Bank (Halyk Group) l ML Engineer, Orange (France Telecom)

Article information:
Manuscript received: 21 Apr 2024; **Accepted:** 22 May 2024; **Published:** 23 Jun 2025

Abstract: In the field of natural language processing (NLP) and large language models (LLMS), a comprehensive study has been conducted on a number of machine learning models and methods for extracting, understanding, and generating textual information. The main focus was on using the capabilities of models of different generations to solve a classification problem using machine learning, which was trained, validated and tested on synthetically generated text data from ChatGPT the GPT-4o version.

The experiments began by evaluating basic models such as TF-IDF, Word2Vec, Doc2Vec, FastText, and ended up with Transformer-based models. These models were used to extract embeddings from text data, which laid the foundation for subsequent intent classification. This initial stage of the experiments allowed to obtain valuable information about the effectiveness of various embedding methods and their impact on the subsequent Chatbot classification task. This article describes an optimization through these various stages of experimentation and learning, culminating in a holistic understanding of the various NLP and LLM models and their applications.

1. Introduction

Text data preprocessing is the basis for analyzing any NLP task. This data is presented in a form that is understandable by machine learning and deep learning algorithms, and the further process is carried out in the form of word or document embeddings, which are vectors of numerical entries. These embeddings contain syntactic and semantic information about symbols, words, or sentences in the text. Machines cannot understand string (text) data the way humans do, and therefore they are provided with representations of text – embeddings that are vectors or arrays containing numeric values, which are machine language.

From TF-IDF to BERT, text representation has undergone radical changes in recent years and has led to improved performance in many tasks, in particular, thanks to pre-trained versions of modern models. These pre-trained models can be used in many tasks without the need to create and train them completely from scratch. This is a huge advantage, but their versatility is limited. When the subject area and vocabulary are specific, they work worse.

In recent years, the field of text representation has undergone a noticeable evolution, moving from traditional methodologies to advanced methods using the capabilities of neural networks. Statistical models provided a solid foundation for quantifying the importance of words in documents, but the limitations of these methods in capturing context and semantics became apparent with the advent of

neural models.

Neural models have become an innovative approach to placing words in a continuous vector space that captures semantic connections and analogies. They have served as a catalyst for the transition to distributed word representations, revolutionizing the way machines understand the nuances of human language. Further extensions based on the original neural models extended this paradigm to entire documents, providing a holistic understanding that goes beyond individual words.

Similarly, the textual representation has undergone significant changes with the advent of transformer-based models (a type of neural network architecture), which introduced a text understanding model BERT. This huge leap led to bidirectional contextualized embeddings, which led to a breakthrough in the field of context definition and semantics. BERT's pre-training and fine-tuning mechanisms have broken down barriers, providing state-of-the-art performance in solving various tasks related to artificial intelligence. The influence of BERT has generated a number of "pre-trained" models that have enhanced its transformative impact.

2. Background

Language can be defined as a set of rules in which characters are combined and used to convey information. Although perceiving linguistic information in the form of text or speech and making sense of it is an extremely complex cognitive ability of our brain, we are not aware of this, because it comes to us naturally. Language is the basis for natural language processing (NLP), which is an area of artificial intelligence and computer science that uses machine learning and deep learning to enable computers to understand and communicate with human language. Being a vast field, it has given us generative AI and large language models (LLM). In recent years, we have been using NLP more and more often without even realizing it. Online translators, chatbots, typo correctors, etc. are all examples of NLP. Large Language Models (LLM) allow you to understand and generate human language in your field. They mostly learn from a huge amount of data from the Internet and books and require a lot of computational power. Pre-trained models can additionally be used for a particular task by being fine-tuned, which will allow them to adapt to the task or field of activity. The beauty of language models is that they can be integrated with non-language models to perform more complex tasks, such as creating images or audio recordings from a text or voice prompt.

2.2 Embeddings

Representing words and documents in such a way that machines can understand and perform tasks on them is one of the most important aspects of NLP. It was found to be useful to represent them as vectors. These vectors with elements in the form of numbers can represent text in a vector space. The entries represent characteristics of the text corpus, such as the frequency of occurrence of words or any other representation. By storing information about a specific text corpus, they become embeddings, where the location of each embedding in a vector space provides useful information, such as how close words or sentences are semantically and syntactically. Words are ambiguous. For example, the word “light” can mean something not heavy or something not dark. We humans can tell the difference based on the content, just like embeddings. An embedding can be trained based on the text corpus carrying the meaning of the word. If the text referred to a light object, then the embedding in a vector space could have the form [-0.5, 0.2], and if the text referred to a non-dark object, then the embedding could be [0.75, 0.6]. In Figure 1, we can see that both words are located in different places in the vector space, which leads to machines treating them as different instances with different symbolic and syntactic meanings. They could serve as different inputs for a machine learning model.

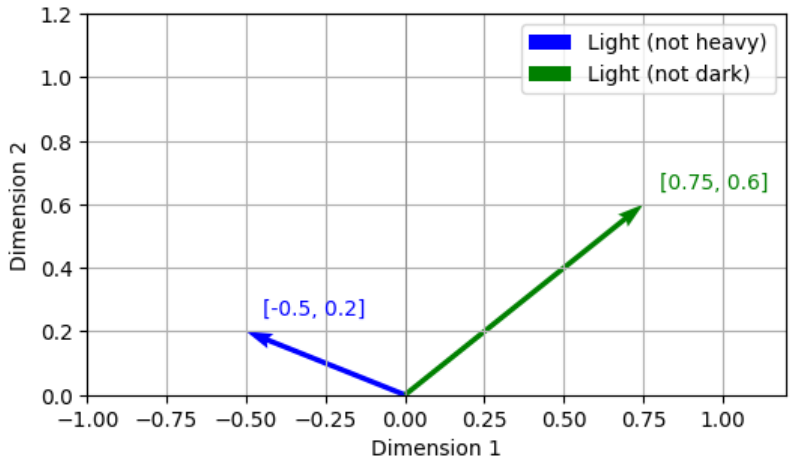

Figure 1: Representaion of word embeddings

2.3 Models

During the survey, two types of machine learning models were used for classification: random forest (RF) and support vector machine (SVM). These are models that have proven their effectiveness in various classification tasks. One of their advantages is the ability to generalize on previously unseen data well.

The random forest algorithm is based on decision trees and ensemble voting. It works well with both numerical and categorical features, reduces overfitting compared to a single decision tree, and handles multidimensional data well. During inference, it outputs the mode of the classes.

$$\hat{y} = \text{mode}\big(h_1(x), h_2(x), \ldots, h_T(x)\big)$$

Random Forest (Classification)

Support Vector Machine is a powerful algorithm that finds the best boundary (called a hyperplane) that separates different classes. The goal is to maximize the margin between the classes — the wider the margin, the better the model can generalize. It operates well with high dimensional data, has strong mathematical and theoretical foundations for classification and is considered effective when the number of features is greater than the number of samples.

$$\hat{y} = \text{sign}(w^T x + b)$$

Support Vector Machine (Linear Classification)

If using a Kernel (for non-linear cases):

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right)$$

Support Vector Machine (With Kernel Function)

Experiments and Results

The main purpose of the experiments conducted in the framework of this study was to compare and optimize modern NLP word embedding techniques that to be used in a classification system of a chatbot. Although many websites currently use artificial intelligence agents in their chatbots, this approach is not necessary, especially for small businesses or early-stage startups. Not to mention that they are quite expensive. Therefore, the use of chatbots, which have a classification model in their system for answering questions or determining intentions, is still in high demand. Before chatbots start processing customer requests, they have their embedding layer that converts the customer's input request in text form into embeddings that the chatbot can understand and process. The quality of a chatbot's

work directly depends on these embeddings, and since different methods produce different embeddings, the experiments showed how well each of them performs. The stages of the experiments were distributed as follows:

- Initial handling of the environment was carried out. That included choosing state-of-the-art models for representation of text, selecting machine learning models for Classification and a set of technologies to be used.
- Text preprocessing which is cleaning and transforming raw text data into a format suitable for embedding models.
- Training the selected embedding models on the generated, by ChatGPT, synthetic data.
- Using the embeddings as inputs and train the Chatbot for classification.

During the preprocessing the text data had gone through the following steps:

1. Tokenization:
- The text was broken down into individual words.
2. Lowercasing:
- All the text was converted into lowercase to ensure uniformity
3. Removing punctuation:
- Punctuation marks such as commas, periods and questions marks were removed.
4. Removing stop words:
- Common words that do not carry much semantic meaning such as articles and some auxiliary verbs were eliminated.
5. Stemming and Lemmatization:
- The words were reduced to their root form.

After the preprocessing step, the text data had been trained with different state-of-the-art models used to retrieve embeddings. For classification the text corpus represented each sentence as a unique question asked the chatbot to reply and their corresponding syntactically created target labels that served as classes to which those questions could be bounded to. The dataset had been split into train/test sets with 80% corresponding to the training set and 20% to the test set.

TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) is a widely used statistical technique in natural language processing and information retrieval. It measures how important a word in a corpus is relative to other words in the corpus by calculating its frequency and proportion. Words that appear less get higher weights than the words that appear more. Although this statistical approach was introduced back in the 1970s, it is still used today for various machine learning and statistical problems.

Term Frequency (TF):

$$TF(t,d) = \frac{\text{Number of occurrences of term } t \text{ in document } d}{\text{Total number of terms in the document } d}$$

Inverse Document Frequency (IDF):

$$IDF(t,D) = \log_e\left(\frac{\text{Total number of documents in the corpus}}{\text{Number of documents with term } t \text{ in them}}\right)$$

TF-IDF:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

TF-IDF calculation

During the experiment, several n-gram ranges of TF-IDF had been explored. The N-Gram range allows to set the sequence length of consecutive words in a given text. For instance, the n-gram range (1, 4) means the algorithm will consider the unigram (only single word), bigram (group of 2 consecutive words), trigram (group of 3 consecutive words) and fourgram (group of 4 consecutive words) to build a vocabulary. Eventually the range of (1, 1) was decided to be used. The sentence embeddings derived, after the model had been trained, had the dimension of 2685, corresponding to the size of the vocabulary. This number represents the number of unique words that the TF-IDF model had encountered. Hence, TF-IDF produced sparse vectors (small number of non-zero elements) in contrast to dense vectors in the later experiments. Further, the derived embeddings were used to train both Random Forest and Support Vector Machine models to classify the sentence vectors into their corresponding classes.

Model	Accuracy	F1-score
Random Forest	0.90	0.85
Support Vector Machine	0.88	0.83

Table 1: Classification performance using TF-IDF embeddings on the test dataset

Despite the embeddings being sparse the F1-score and Accuracy demonstrated high values.

Word2Vec

The next approach for obtaining word embeddings came much later, in 2013, and it is called Word2Vec. It was introduced by Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean at Google in their research paper titled "Efficient Estimation of Word Representations in Vector Space. As the name suggests, it takes a word and transforms it into dense vectors. Unlike the previous approach, this uses Neural Networks to train embeddings. It has two types of architecture:

- Continuous bag-of-words (CBOW)
- Continuous skip-gram.

Continuous bag-of-words (CBOW) predicts the center word from the unstructured raw input text in a fixed context window size and Continuous skip-gram does the opposite, trying to predict the context (surrounding) words. This technique is called self-supervised learning as it combines both supervised and unsupervised learning. The initial training text corpus is raw and has no labels, however during training, the target words that the model tries to predict become the labels. Both architectures use 2 neural layers. This approach allows embeddings to capture semantic and syntactic relationships between the words. The Gensim library was used to train word embeddings. Those word embeddings had been then aggregated into sentence embeddings with the MeanPooling technique. With MeanPooling every word embedding gets their element-wise mean and concatenates into sentence embeddings. Table 2 gives the results of the Random Forest and Support Vector Machine classifiers with the derived sentence embeddings.

Model	Accuracy	F1-score
Random Forest	0.89	0.82
Support Vector Machine	0.452	0.42

Table 2: Classification performance using Word2Vec embeddings on the test dataset

Although, Word2vec is considered a newer approach that uses neural networks to train dense vectors, it did poorly in the classification task with Support Vector Machine compared to TF-IDF.

Doc2Vec

Doc2Vec, also known as Paragraph Vector, is an extension of the Word2Vec model designed to generate dense vector representations for entire documents or paragraphs of text. It was proposed by Le and Mikolov in their paper "Distributed Representations of Sentences and Documents" in 2014. Similar to Word2Vec, Doc2Vec uses neural network architectures- Continuous Bag of Words (CBOW) and Skip-gram to train dense document embeddings. The key idea is to include an additional vector, called a document tag, which is used to represent the document context along with the word context. This approach needs no Mean Pooling on a sequence of words as it generates vectors representing the syntactic and semantic meaning of the whole sentence (document). The CBOW architecture was used to train the embeddings.

Model	Accuracy	F1-score
Random Forest	0.766	0.78
Support Vector Machine	0.689	0.70

Table 3: Classification performance using Doc2Vec embeddings on the test dataset

The results of Doc2vec do not differ much from that of its predecessor Word2Vec. However, Doc2Vec is extremely convenient working with sentences instead of words as it automatically trains document (sentence) embeddings.

FastText

FastText is an open-source library for efficient text representation and classification developed by Facebook's AI Research (FAIR) lab. The research paper is titled "Enriching Word Vectors with Subword Information" by Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, which waspublishedin2016. It is an extension of Word2Vec but unlike it, it represents each word in a character level forming groups of characters using the skip-gram technique. It handles out-of-vocabulary (OOV) words by constructing words out of symbols. As a result, FastText captures the semantic and syntactic meaning of words even for rare, incorrect or unseen words.

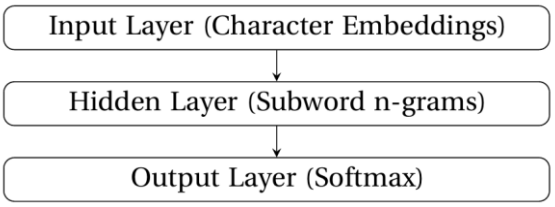

Figure 2: FastText Character-Level Word Model

Character-level words:

"fast" → ["fa", "as", "st"]

"text" → ["te", "ex", "xt"]

Model	Accuracy	F1-score
Random Forest	0.781	0.75
Support Vector Machine	0.623	0.60

Table 4: Classification performance using FastText embeddings on the test dataset

With FastText one does not have to worry about the unseen words in the test dataset that the trained model's vocabulary had not included. This can also serve as a feature of stability.

BERT

The paper called "Attention is all you need" by Google scholars and the University of Toronto first introduced transformers and moved forward the field of AI both in academia and industry, and made NLP extremely popular among machine learning engineers. The transformer is a deep

learning architecture based on the multi-head attention mechanism, that uses neural networks to train embeddings from text. The transformer's encoder part of the architecture led to the development of a model called BERT (Bidirectional Encoder Representations from Transformers) that was introduced by Jacob Devlin and his colleagues at Google AI Language. It was released in 2018. The paper detailing the BERT model is titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. BERT trains deep bidirectional embeddings from unlabeled text by jointly learning on both left and right context. One of the biggest advantages of BERT and other transformer-based models is the ability to be fine-tuned. This way the models can perform a desired task without the need to be fully trained from the beginning. It greatly reduces the computation time, computational power and helps outperform other models. Pre-trained BERT did not perform that well as can be observed in table 5.

Model	Accuracy	F1-score
Random Forest	0.66	0.62
Support Vector Machine	0.63	0.58

Table 5: Classification performance using pre-trained BERT embeddings on the test dataset

However, after fine-tuning the BERT model the result outperform all the previous models.

Model	Accuracy	F1-score
Random Forest	0.95	0.87
Support Vector Machine	0.92	0.84

Table 6: Classification performance using fine-tuned BERT embeddings on the test dataset

Conclusion

This study provided a comprehensive analysis and optimization of various Natural Language Processing (NLP) embedding methods and their performance in the supervised intent classification task for a chatbot system. Beginning with traditional statistical technique such as TF-IDF and advancing through neural network-based methods like Word2Vec, Doc2Vec, FastText, and finally, Transformer-based models as BERT, the experiments showed the strength of each of them. The results say that TF-IDF is a great technique despite being the oldest among them. The fine-tuned BERT proved the power of the Transformer by having demonstrated the highest performance. The results show that lighter embedding methods for resource-constrained systems still can be a great option. If resources allow then Large language Models are the best to achieve the highest performance possible.

References

1. Felipe Almeida and Geraldo Xexéo. *Word Embeddings: A Survey*. 2023. arXiv: 1901.09069 [cs.CL].
2. Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. 2017. arXiv: 1607.04606 [cs.CL].
3. Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A: 1010933404324. URL: https://doi.org/10.1023/A:1010933404324.
4. R. J. G. B. Campello et al. "Density-based clustering based on hierarchical density estimates". In: *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'11)* 8158 (2013), pp. 160–172. DOI: 10.1007/978-3-642-40994-3_14.
5. Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018.
6. Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.
7. Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language*

Understanding. 2019. arXiv: 1810.04805 [cs.CL].

8. Divyansh Khurana et al. “Natural Language Processing: State of the Art, Current Trends and Challenges”. In: *Multimedia Tools and Applications* 82 (2023), pp. 3713–3744. DOI: 10.1007/s11042-022-13428-4. URL: https://doi.org/10.1007/s11042-022-13428-4.
9. Quoc V. Le and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. 2014. arXiv: 1405.4053 [cs.CL].
10. *LLAMA: Large Language Models for Articulated Motion and Animation*. https://ai.meta.com/llama/. Accessed: August 1, 2023.
11. Stuart Lloyd. “Least Squares Quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.
12. Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit*. 2002. arXiv: cs/0205028 [cs.CL].
13. lovelytony22. *Why Are There So Many Different Languages?* https://africanparadiseworld.com/2018/01/29/why-many-different-languages/. Accessed on August 10, 2023.
14. Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data Using t-SNE”. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.
15. Andrzej Maćkiewicz and Waldemar Ratajczak. “Principal Components Analysis (PCA)”. In: *Computers & Geosciences* 19.3 (1993), pp. 303–342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004 (93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.
16. James B. MacQueen. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1.14 (1967), pp. 281–297.
17. Louis Martin et al. “CamemBERT: a Tasty French Language Model”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.acl-main.645. URL: https://doi.org/10.18653/v1/2020.acl-main.645.
18. **[18]** Leland McInnes, John Healy, and James Melville. “UMAP: Uniform Manifold Approximation and Projection”. In: *Journal of Open Source Software* 3.29 (2018), p. 861. DOI: 10.21105/joss.00861.
19. Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].
20. OpenAI. *ChatGPT*. https://openai.com. 2021.
21. OpenAI. *OpenAI*. 2023. URL: https://www.openai.com/.
22. *Orange Group: Overview*. https://www.orange.com/en/group/overview/orange-group. Accessed: July 18, 2023.
23. Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2018. arXiv: 1201.0490 [cs.LG].
24. Radim Řehůřek. *Gensim: Topic Modelling for Humans*. https://radimrehurek.com/gensim/index.html. Accessed on: August 3, 2023.
25. Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT Networks”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020).

26. Ippon Technologies. “Mettez en place une classification de texte performante grâce à un CamemBERT”. In: *Ippon Blog* (Oct. 2022). URL: https://blog.ippon.fr/2022/10/10/mettez-en-place-une-classification-de-texte-performante-grace-a-un-camembert-2/.
27. *TF-IDF (Term Frequency-Inverse Document Frequency)*. https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/. Accessed: July 2023.
28. Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].
29. IBM. *Natural Language Processing*. https://www.ibm.com/think/topics/natural-language-processing. Accessed: August 1, 2023.