Evaluating Semantic Similarity Search: A Methodological Approach to Comparing Language Models and Algorithms

ABSTRACT

This research paper introduces a novel methodology for the comparative analysis of various Language Models (LMs) and semantic similarity algorithms in the context of information retrieval. Our study aims to introduce a standardized method to evaluate and compare models based on their semantic similarity search capabilities that comprise quantitative measures. The LMs used include Word2Vec, MiniLM, and RoBERTa, evaluated with similarity metrics such as Cosine Similarity, Euclidean Similarity, and Dot Product Similarity. The methodology also considers different storage methods, specifically local storage and Pinecone Index, and employs the precision@k methodology with k set to 3. The results are analyzed using a comprehensive scoring formula, considering time taken for retrieval, precision score, and similarity metric score. We introduce a novel evaluation metric, Language Model Assessment for Semantic Similarity (ReLMASS). These findings contribute to the broader field of Natural Language Processing and Conversational AI, offering insights into the optimal combinations for efficient and accurate information retrieval.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; **Natural language processing**; • **Information systems → Evaluation of retrieval results**.

KEYWORDS

Information Retrieval, Large Language Models, Natural Language Processing, Pre-trained models, AI Applications, Machine Learning, Chatbots, Evaluation Metrics

ACM Reference Format:

. 2023. Evaluating Semantic Similarity Search: A Methodological Approach to Comparing Language Models and Algorithms. In *Proceedings of ACM SIGIR Conference on Human Information Interaction and Retrieval (ACM SIGIR CHIIR '24).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The vast expansion of unstructured data and the growing need for meaningful interactions between humans and machines have necessitated more refined techniques in information retrieval and natural language processing. In the realm of Natural Language Processing (NLP), the ability to accurately retrieve information based on semantic similarity is a critical task. This involves identifying and retrieving sentences from a text corpus that are semantically similar to a given query sentence. The effectiveness of this process is determined by the choice of Language Models (LMs) and semantic similarity algorithms used.

This research paper embarks on a comprehensive comparative study of various LMs and semantic similarity algorithms to identify a standard approach to evaluate combinations for semantic similarity search. The LMs under consideration include Word2Vec, MiniLM, and RoBERTa, which are among the most advanced models currently used in NLP. These models are evaluated in combination with different similarity metrics, namely Cosine Similarity, Euclidean Similarity, and Dot Product similarity. This study investigates the following research questions:

(1) How does the effectiveness of different comparable language models vary across similarity metrics?
(2) To what extent does the similarity scores provided by the models align with the actual similarity of the two sentences?

The study also explores the impact of different storage methods on the performance of these combinations. Specifically, it compares the use of local storage and Pinecone Index, a vector database designed for efficient information retrieval. The performance of each combination is evaluated using the precision@k methodology, with k set to 3. This means that the top 3 sentences retrieved by each combination are used to compute the precision@3 score, which forms the basis for ranking the combinations using the comprehensive scores.

The unique aspect of this study is the incorporation of the similarity metric score along with the precision@k score into the final comprehensive score. This is achieved through a formula that takes into account the similarity metric score only if the retrieved sentence is semantically similar to the query sentence. The ultimate goal of this study is to provide a comprehensive evaluation strategy towards understanding the performance of different combinations of LMs and similarity algorithms in semantic similarity search. The findings of this study will contribute to the broader field of NLP and conversational AI, providing valuable insights for researchers and practitioners alike.

We focus on the methodology and the comparative aspect of different models and algorithms, and aim to address a gap in semantic similarity evaluation research. While models are becoming more sophisticated, there has not been a significant move to quantify their effectiveness that also includes metrics beyond those provided by the software. Our approach incorporates metrics such as p@k along with the **Cumulative Model-Specific Semantic Similarity Score (CMSSS)**. This provides researchers with tools to design more user-centered systems. Our study offers a robust framework that can be applied to various models and domains. The methodology's comprehensive nature ensures a nuanced and standardized approach, paving the way for further research and potential advancements in the field of semantic search evaluation

ACM SIGIR CHIIR '24, March 10–14, 2024, Sheffield, UK

© 2018 Association for Computing Machinery.

ACM ISBN XXX-X-XXXX-XXXX-X/XX/XX…$XX.XX

https://doi.org/XXXXXXX.XXXXXXX

and comparison.

2 LITERATURE REVIEW

The field of Natural Language Processing (NLP) has seen significant advancements in recent years, particularly in the area of semantic similarity. Semantic similarity is a measure of the degree to which two pieces of text carry the same meaning. This concept is critical in various NLP tasks such as information retrieval, text classification, question answering, and plagiarism detection.

2.1 Semantic Similarity

Several surveys exist in the literature regarding semantic similarity that explore the various methods and applications that have been developed. Chandrasekaran et al.[1] provide a comprehensive overview of the concept of semantic similarity and trace its evolution over time. Their work discusses various methods and techniques used to measure semantic similarity, including knowledge-based methods, corpus-based methods, and hybrid methods. Their work highlights the challenges and research problems in estimating semantic similarity in NLP. However, they do not delve into the practical implementations of these methods in real-world scenarios.

Sunilkumar et al.[8] explore the relevance of semantic similarity in NLP. Their work discusses the applications of semantic similarity in fields such as information retrieval, text classification, question answering, and plagiarism detection and provide an overview of the different methods used to measure it. However, they do not compare the performance of different language models in these tasks.

Zhang et al. [13] provide a comprehensive framework BERTScore to evaluate the semantic similarity between a candidate sentence and a given sentence. They achieve this by calculating recall, precision, and F1 score by using BERT [2] embeddings. Their method does not use any thresholding or p@k metrics that are important for a number of downstream task in most NLP applications being used.

Hernández-Farías et al. [5] study Semantic Textual Similarity (STS), specifically in the context of social media. Their work introduces a new method to detect irony in tweets by focusing on the sentiment ingrained within them. One limitation of their method is that it depends on sentiment analysis resources, which may not exist for all languages.

The present study contributes to the assessment of how different models exhibit different performance metrics when storages are changed and how the scores can play a role in deciding the implementation based on the needs.

2.2 Methodological Improvements and Proposals

Zhang et al. [12] discusses the challenges of extracting text in a fine-grained way in NLP. They propose a combination of methods, including word embedding, semantic role labeling, and semantic similarity computing, to overcome these challenges. Their work provides a detailed analysis of the proposed methods and their effectiveness in improving the accuracy of semantic similarity computation. However, they do not evaluate the performance of these methods in terms of precision and time efficiency, which is a key focus of our study.

Sun et al. [10] discuss the task of quantitatively measuring the semantic relatedness between two sentences. The authors propose a method that considers the context of the sentences to improve the accuracy of semantic similarity measurement. This study provides a detailed analysis of the proposed method and its effectiveness in various NLP tasks.

Rahutomo et al. [9] propose an enhancement of Cosine Similarity measurement by incorporating semantic checking between dimensions of two term vectors. Their work provides a detailed analysis of the proposed method and its effectiveness in improving the accuracy of semantic similarity measurement.

2.3 Applications and Domain-Specific Studies

Esteva et al.[3] present a semantic, multi-stage search engine tailored for the COVID-19 literature, highlighting the potential of semantic similarity in specialized domains. The authors use deep learning techniques to extract semantic information from the text and use it to improve the accuracy of information retrieval. The paper provides valuable insights into the application of semantic similarity searches in a specific domain.

Masuda et al. [7] introduce a framework for semantic information retrieval based on the integration of various NLP techniques. They discuss the challenges of semantic search and propose a method that combines different NLP techniques to overcome these challenges. Their work paper provides a valuable perspective on semantic search; however, it does not evaluate their performance.

While the existing literature offers in-depth insights into semantic similarity searches and its various applications, there is a noticeable gap in the practical evaluation of different language models and similarity metrics. Many studies, despite their thorough reviews of existing and developing methods, have not assessed performance metrics in terms of precision and time efficiency in real-world scenarios. To bridge this gap, our study tries to emphasize and address the lack of quantitative metrics by applying and evaluating these methods on a specific dataset. We implement a workflow for evaluating and comparing models on their semantic similarity search capabilities.

3 METHOD

This section provides a detailed explanation of the methodology employed in this research. The research process involved the creation of a standard evaluation process, use of three different language models ranging from a small-sized and baseline model to a large-sized and complex language model, three similarity metrics, and two storage methods. The methodology also includes the preparation of the corpus and query sentences, the execution of the experiment, and the evaluation of the results.

3.1 Similarity Metrics, and Storage Methods

Three different similarity metrics were used to measure the semantic similarity between the query sentences and the sentences in the corpus: Cosine Similarity, Euclidean Distance, and Dot Product. These metrics were chosen because they are commonly used in information retrieval and have been shown to be effective in measuring semantic similarity.

Two different storage methods were used: Pinecone, a vector database service, and local storage. The use of Pinecone allowed for efficient information retrieval, while local storage served as a baseline for comparison.

3.2 Model Selection and Application

We employ various language models to encode our corpus and query sentences into embeddings. Three different language model frameworks were used in this research: Word2Vec, MiniLM[11], and RoBERTa [6]. These models were selected based on their sizes and complexity, with RoBERTa being the largest and the most complex, followed by MiniLM, and then Word2Vec. Each of these models has been proven effective in various natural language processing tasks, and their use in this research allowed for a comprehensive comparison of their performance in semantic similarity search. These models have been chosen based on our literature review and include the following:

1. Word2Vec: As a baseline, we use the Word2Vec model to analyze the semantic proximity between sentences. Word2Vec's simplicity and fundamental approach provide a necessary comparison for the more complex transformer models.
2. MiniLM (all-MiniLM-L6-v2): This model is selected for its size efficiency and competitive performance. MiniLM provides a balance between resource usage and semantic understanding.
3. RoBERTa (all-roberta-large-v1): Recognized for its robust performance, the RoBERTa model is utilized as the biggest and most complex of the models in our study

We set up a script that utilizes three models for our study, as described in the method section. The script focuses on calculating the similarity between one sentence and others using embeddings from Word2Vec [4] for the semantic searches. The parts of the script utilizing MiniLM and RoBERTa leverage the models described to encode and query sentences; they also calculate similarity metrics, and storage methods to estimate semantic similarity.

3.3 Corpus Preparation and Query Generation

The corpus used in this research was a text file containing sentences about Barack Obama. The corpus was split into individual sentences, and each sentence was tokenized using the Spacy tokenizer.

The query sentences were generated by slightly modifying certain sentences from the corpus. These sentences were chosen because that they had a high semantic similarity with at least one sentence in the corpus. The query sentences were also tokenized using the Spacy tokenizer.

The original sentence from the dataset was: "In 2008, a year after beginning his campaign, and after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president." The input text to the models was: "In the year 2008, after a tightly contested primary against Hillary Clinton and following the start of his political journey a year earlier, the Democratic Party nominated him for the presidency."

3.4 Experiment Process

For each combination of model, similarity metric, and storage method, the following steps were performed:

- The model was used to generate embeddings for the sentences in the corpus and the query sentences.
- If Pinecone was used as the storage method, the corpus embeddings were added to a Pinecone index.
- The similarity between each query sentence and the sentences in the corpus was calculated using the chosen similarity metric.
- The top 3 most similar sentences for each query were retrieved, and their similarity scores were recorded.

The time taken to retrieve the most similar sentences was also recorded for each combination, but it was not included in the final ranking score, which is the comprehensive similarity score.

Given a corpus C and a set of query sentences Q, we aim to compute a similarity score S.

$$T(C) \text{ is the tokenization of } C$$
$$T(Q) \text{ is the tokenization of } Q$$
$$E(T) \text{ is the embedding of tokens } T$$
$$S_{M,SM,SMET} = f(E(T(Q)), E(T(C)))$$

Where M is the chosen embedding model, SM is the storage method, $SMET$ is the similarity metric, and f is a function that computes similarity based on the chosen metric.

Algorithm 1 Semantic Similarity Scoring: Language Model Assessment for Semantic Similarity (ReLMASS)

```
procedure EvaluateSimilarity(corpus, query_sentences)
    Initialize tokenizer
    Parse the corpus into corpus_sentences
    Tokenize query_sentences
    for model in embedding_models do
        Compute embeddings for corpus_sentences and query_sentences using model
        for metric in similarity_metrics do
            Compute similarity scores based on metric
            for query_embedding in query_embeddings do
                Retrieve most similar sentences
                Compute the CMSSS
                Compute precision@k for retrieved sentences
                ReLMASS score = $\frac{(\sum \text{CMSSS})\cdot\text{precision@k}}{k}$
                Print ReLMASS score
```

3.5 Evaluation

The performance of each combination of model, similarity metric, and storage method was evaluated using the precision at k ranking methodology, with k set to 3. The algorithm returns the top 3 most similar sentences retrieved for each query, which were then used to calculate the precision score.

A matched sentence was considered semantically similar if its similarity score was greater than 0.5 and if it was in fact semantically similar after human verification. The matched would then be given a precision score by dividing the number of truly semantically similar sentences by k (0.333 per correctly matched sentence). Semantic similarity is also examined by an end user, since it is important to be able to distinguish between results that are reasonably similar and results are completely wrong but are presented as similar results due to the models being unable distinguish between. If a result is completely unrelated, it will be assigned a precision score of 0. This part of the evaluation is done manually as the scope of our study is to examine the metrics affecting searches rather than building more accurate search methods. This implies that a model exhibiting a precision score of 0.999 has demonstrated optimal performance, a score of 0.666 indicates moderate performance, a score of 0.333 suggests marginal performance, and a score of 0 signifies poor performance.We intentionally use simple sentences within a corpus to make the evaluation points more distinct.

The final ranking score for each combination was calculated using the novel formula:

$$\text{ReLMASS score} = \frac{(\sum \text{CMSSS}) \cdot \text{precision@k}}{k} \tag{1}$$

Where,

- CMSSS is Cumulative Model-Specific Semantic Similarity Score and measures the correct matched sentence similarity score
- k is the number of retrieved matched sentences being compared

This ReLMASS score, referred to as the comprehensive score, factors in the semantic similarity scores, precision scores, and the value of k. It provides a holistic measure of the performance of each combination, taking into account both the quality of the matched sentences (as indicated by the semantic similarity scores) and the quantity of correct matches (as indicated by the precision scores).

3.6 Results Comparison

The results of the experiment were compared and analyzed based on the comprehensive scores of the different combinations. These scores provided a measure of how well each combination performed in terms of semantic similarity search, considering both the quality and quantity of the matched sentences.

The results comparison allowed for a comprehensive evaluation of the performance of the different combinations of models, storage methods, and similarity metrics. This evaluation provided insights into the effectiveness of these combinations in semantic similarity search, and it formed the basis for the conclusions drawn in this study.

4 RESULTS AND ANALYSIS

The experiment was conducted using three different language model frameworks (Word2Vec, MiniLM, and RoBERTa), three similarity metrics (Cosine Similarity, Euclidean Similarity, and Dot Product Similarity), and two storage methods (Local and Pinecone Index). The results were evaluated using the comprehensive score introduced in Method sections which uses similarity scores and the precision at k ranking methodology, with k set to 3. The K value indicates the number of "top *k*" matched sentences by the combination of model, storage method, and similarity metric. This is then used to were used to create the precision@K score which was used in ranking them.

The results of the experiment are presented below in a series of tables and graphs. These results provide a comprehensive overview of the performance of each combination of model, similarity metric, and storage method.

For the purpose of this paper, we have shown the results from the ReLMASS method for only k=3. During our experiments we got comparable results after changing the K values to 5,7.

4.1 ReLMASS scores and similarity metric comparison with respect to storage methods

The plot in Fig1 shows the comprehensive scores of each model and each similarity metric using different storage methods. This score takes into account the semantic similarity scores, precision scores, and the value of k, providing a holistic measure of the performance of each combination.

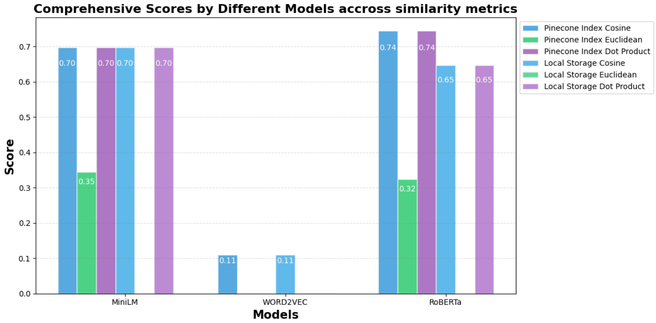

Figure 1: ReLMASS Score

4.2 Time Comparison across models and similarity metrics

The graph in Fig2 compares the time taken by the three models across all three similarity metrics under Pinecone storage vs local storage. This comparison of the efficiency of the storage methods in terms of retrieval speed is an important consideration when designing system for scalability.

The plot in Fig3 shows the average time taken by the algorithm. From the two plots, it is quite clear that RoBERTa is the fastest of the methods being studied.

4.3 Model Performance Results

The experimental results presented in the tables below provide a comprehensive overview of the performance of each combination of model, similarity metric, and storage method. These tables

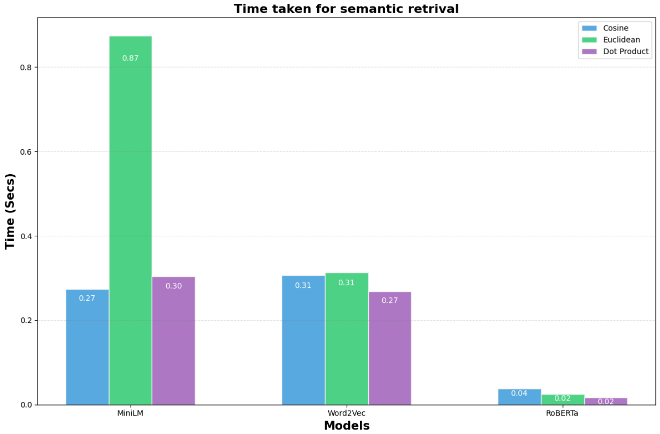

Figure 2: Time Taken to match and retrieve sentences

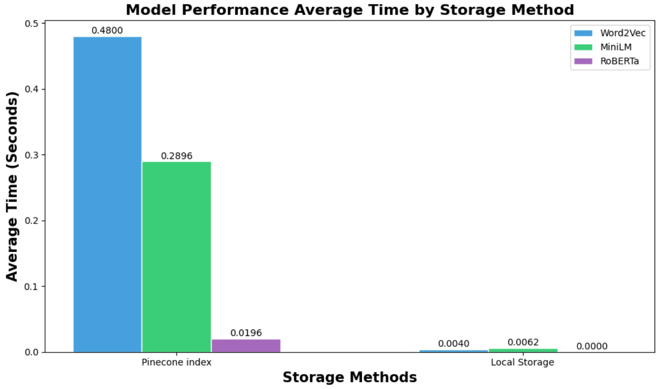

Figure 3: Average Time Taken to match and retrieve

encompass the similarity scores and the time taken for each evaluation.

4.3.1 *Word2Vec Model Performance.*

Table 1 shows the statistics of the Word2Vec model. This examines two storage methods—Pinecone Index and Local Storage—and evaluates their performance with three similarity metrics: Cosine, Euclidean, and Dot Product.

Table 1: Word2Vec Model Performance

Storage Method	Similarity Metric	Time (Secs)	ReLMASS Score
Pinecone Index	Cosine	0.2943	0.1098
Pinecone Index	Euclidean	0.3074	0
Pinecone Index	Dot Product	0.2673	0
Local Storage	Cosine	0.0121	0.1098
Local Storage	Euclidean	0.0054	0
Local Storage	Dot Product	0.0011	0

4.3.2 *MiniLM Model Performance.*

Table 2 shows the performance metrics using the MiniLM model. Similar to the previous table, we test the model's performance against both the Pinecone Index and Local Storage, using the same three similarity metrics.

Table 2: MiniLM Model Performance

Storage Method	Similarity Metric	Time (Secs)	ReLMASS Score
Pinecone Index	Cosine	0.2713	0.6972
Pinecone Index	Euclidean	0.8683	0.3451
Pinecone Index	Dot Product	0.3004	0.6972
Local Storage	Cosine	0.0028	0.6972
Local Storage	Euclidean	0.0054	0
Local Storage	Dot Product	0.0036	0.6972

4.3.3 *RoBERTa Model Performance.*

As with the previous two models, Table 3 displays the performance metrics for the RoBERTa model.

Table 3: RoBERTa Model Performance

Storage Method	Similarity Metric	Time (Secs)	ReLMASS Score
Pinecone Index	Cosine	0.0315	0.7449
Pinecone Index	Euclidean	0.0134	0.3234
Pinecone Index	Dot Product	0.0139	0.7449
Local Storage	Cosine	0.0057	0.6479
Local Storage	Euclidean	0.0113	0
Local Storage	Dot Product	0.0028	0.6479

4.4 Ranking of Combinations

Based on the final comprehensive score for each combination of model, storage method, and similarity metric, the ranking table would be as shown in Table 4:

4.5 Analysis

The results of the experiment show that the RoBERTa model combined with the Pinecone storage method and either the Cosine Similarity or Dot Product Similarity metric achieved the highest comprehensive scores, indicating the best performance in terms of semantic similarity search. The MiniLM model also performed well, especially when combined with the Cosine Similarity or Dot Product Similarity metric and either the Pinecone storage method or local storage.

On the other hand, the Word2Vec model achieved the lowest comprehensive scores, indicating the poorest performance. This suggests that the Word2Vec model may not be the best choice for semantic similarity search, especially when compared to more complex models like RoBERTa and MiniLM.

In terms of storage methods, the Pinecone storage method generally achieved higher comprehensive scores than local storage, especially when combined with the RoBERTa or MiniLM model

Table 4: Ranked Model Performance

Rank	Model	Storage	Metric	ReLMASS Score
1	RoBERTa	Pinecone Index	Cosine	0.7449
2	RoBERTa	Pinecone Index	Dot Product	0.7449
3	MiniLM	Pinecone Index	Cosine	0.6972
4	MiniLM	Pinecone Index	Dot Product	0.6972
5	MiniLM	Local Storage	Cosine	0.6972
6	MiniLM	Local Storage	Dot Product	0.6972
7	RoBERTa	Local Storage	Cosine	0.6479
8	RoBERTa	Local Storage	Dot Product	0.6479
9	RoBERTa	Pinecone Index	Euclidean	0.3234
10	MiniLM	Pinecone Index	Euclidean	0.3451
11	Word2Vec	Pinecone Index	Cosine	0.1098
12	Word2Vec	Local Storage	Cosine	0.1098
13–18	Various	Various	Various	0

and either the Cosine Similarity or Dot Product Similarity metric. However, the Pinecone storage method also took longer to evaluate the matching, indicating a trade-off between performance and efficiency.

Overall, these results provide valuable insights into the performance of different combinations of language models, similarity metrics, and storage methods in semantic similarity search. They highlight the importance of choosing the right combination to achieve the best performance, and they provide a basis for further research and optimization in this area.

5 DISCUSSION

5.1 Emphasis on Evaluation Methodology

The core contribution of this research lies in the development of a standard approach towards evaluating language models for semantic similarity searches. The comprehensive evaluation methodology, characterized by the novel formula for calculating the comprehensive score, sets a new benchmark in the field. This formula, which integrates semantic similarity scores, precision scores, and the value of k, offers a holistic measure of performance, reflecting both the quality and quantity of matched sentences. We changed the values of K to 5 and 7 and got comparable results.

Using all-MiniLM-L6-v2 model with no storage index and used cosine Similarity as the similarity metric, the algorithm gives us the scores below:

(1) ReLMASS score with K = 5 is 0.5362
(2) ReLMASS score with K = 7 is 0.4571

Our results reveal significant insights into the performance of different language models, similarity metrics, and storage methods. These findings underscore the importance of selecting the right combination of models and metrics to achieve optimal performance in semantic similarity search. Projects that involve building custom AI chatbots will be able to use evaluation metrics like ours while designing algorithms that empahsise on the more interesting first few results, that return highly relevant results in those top positions. Our metric that utilises Precision@k directly measures how well the model is performing in this aspect while incorporating a form of intrinsci evaluation.

5.2 Unexpected Findings and Implications

The unexpected effectiveness of the Pinecone storage method and the trade-off between performance and efficiency highlight the complexities of semantic similarity search methods. These insights, derived from scores generated from the ReLMASS algorithm, contribute valuable guidance for researchers and practitioners.

5.3 Comparison with Existing Literature

The research's focus on creating a standardized evaluation approach both affirms and extends existing knowledge. The innovative methodology enriches the current discourse on semantic similarity search, positioning this study as a significant contribution to the field.

6 LIMITATIONS AND FUTURE WORK

The research opens several promising avenues for future exploration:

(1) Refinement of Evaluation Methodology: The current evaluation method could be further refined and standardized, potentially becoming a widely accepted benchmark in the field.
(2) Inclusion of Time in Comprehensive Formula: Future work could explore the integration of time taken for search into the comprehensive formula by assigning it an appropriate weight. This addition would provide a more nuanced understanding of efficiency and performance trade-offs.
(3) Expansion of Models and Metrics: Investigating additional models, metrics, or storage methods could lead to more comprehensive insights.
(4) Development of Adaptive Frameworks: Creating frameworks that dynamically select the best combination based on specific requirements may offer more tailored solutions.
(5) One major limitation is that our methodology incorporates a Natural Language Interpreter in the form of a human user to ensure that a matched sentence is actually relevant to the candidate sentence. This would affect that scalability of the algorithm. In the future, a sufficiently capable LLM could be incorporated as an interpreter to support the scalability.

7 CONCLUSION

This research represents a significant advancement in the field of semantic similarity search evaluation, with its primary focus on creating a standard approach towards evaluating language models. The innovative evaluation methodology, characterized by the comprehensive score formula, sets a new standard in the field, offering a holistic and nuanced understanding of performance.

The findings of the study, the novel insights, and the future directions all contribute to a richer understanding of semantic similarity search. The research not only provides valuable guidance for current practitioners but also lays a strong foundation for future innovations, promising continued growth and refinement in this vital area of NLP.

REFERENCES

[1] Chandrasekaran, D., and Mago, V. Evolution of semantic similarity—a survey. *ACM Comput. Surv. 54*, 2 (feb 2021).

[2] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Esteva, A., Kale, A., Paulus, R., Hashimoto, K., Yin, W., Radev, D., and Socher, R. Covid-19 information retrieval with deep-learning based semantic search, question answering, and abstractive summarization. *NPJ digital medicine 4*, 1 (2021), 68.

[4] Gomez-Perez, J. M., Denaux, R., Garcia-Silva, A., Gomez-Perez, J. M., Denaux, R., and Garcia-Silva, A. Understanding word embeddings and language models. *A Practical Guide to Hybrid Natural Language Processing: Combining Neural Models and Knowledge Graphs for NLP* (2020), 17–31.

[5] Hernández-Farías, I., Benedí, J.-M., and Rosso, P. Applying basic features from sentiment analysis for automatic irony detection. In *Pattern Recognition and Image Analysis: 7th Iberian Conference, IbPRIA 2015, Santiago de Compostela, Spain, June 17-19, 2015, Proceedings 7* (2015), Springer, pp. 337–344.

[6] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019.

[7] Masuda, K., Matsuzaki, T., and Tsujii, J. Semantic search based on the online integration of nlp techniques. *Procedia-Social and Behavioral Sciences 27* (2011), 281–290.

[8] P., S., and Shaji, A. P. A survey on semantic similarity. In *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)* (2019), pp. 1–8.

[9] Rahutomo, F., Kitasuka, T., and Aritsugi, M. Semantic cosine similarity.

[10] Sun, X., Meng, Y., Ao, X., Wu, F., Zhang, T., Li, J., and Fan, C. Sentence similarity based on contexts. *Transactions of the Association for Computational Linguistics 10* (2022), 573–588.

[11] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[12] Zhang, P., Huang, X., Wang, Y., Jiang, C., He, S., and Wang, H. Semantic similarity computing model based on multi model fine-grained nonlinear fusion. *CoRR abs/2202.02476* (2022).

[13] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. Bertscore: Evaluating text generation with bert, 2020.