Semantic Similarity Analysis of Textual Data

Senthil Arumugam  
  
senthil.ramasamy@stud.fra-uas.de

Neetu Ninan

[neetu.ninan@stud.fra-uas.de](mailto:neetu.ninan@stud.fra-uas.de)

*Abstract— in the era of big data and natural language processing, the ability to accurately analyze and compare textual data is paramount. This research presents a comprehensive framework for semantic analysis of textual data, focusing on the calculation of phrase similarities and document comparisons. Leveraging advanced open AI embedding techniques and Cosine Similarity Algorithms, our approach aims to analyses the accuracy and efficiency of semantic similarity assessments and understands its potential use case applications. The framework is implemented as a software tool that preprocesses textual data, generates embedding’s using state-of-the-art models, and calculates similarity scores between phrases and documents. The tool supports various preprocessing options, including tokenization, normalization, and context-based adjustments, ensuring robust and contextually relevant similarity measurements. We evaluate the performance of our framework through extensive experiments on diverse datasets, demonstrating its effectiveness in capturing semantic nuances and improving the quality of textual data analysis. To visualize the effectiveness of our similarity calculations, we have developed a plotting tool that maps scalar values against similarity scores allowing for a clear and immediate understanding of the data distribution and the performance of our similarity metrics and where secondary plot displays the number of possible comparisons between documents and its corresponding Similarity score which displays the semantic analysis between documents or phrases with its relevance based on pre-defined threshold. The results indicate significant improvements in similarity scoring accuracy compared to traditional methods. This research contributes to the field of natural language processing by providing a scalable and flexible solution for semantic analysis, with potential applications in automated content categorization like resume filtering for relevant job opportunities or filtering admission for students based on admission requirements and other several use cases across different domains. The software tool is made available as an open-source project, encouraging further research and development in this analysis or develops potential use cases. Keywords—Embedding, Cosine Similarity, scalar*

# **Introduction**

# **LITERATURE SURVEY**

# **implementation**

Based on the literature review and concepts of Open AI embedding, Semantic analysis, application was designed in such a way that it supports the methods to do analysis of data using phrases and documents as well; We understood that in order to do such analysis using various datasets such as documents or phrases, analysis results may vary due to contextual meaning of the sentences in the documents, During the Initial Study we tried several samples with Open AI Embedding API’s using Nuget package available in the dotnet also we tried one other tool called as Hugging Face API, we found out that due to the restrictions of usage of its API’s is not available in Nuget package , research started with the sole focus on using technique of open AI Embedding because of its availability and its ability to create the embedding which supports the contextual analysis for different types of datasets such as Phrases/Words or documents comparisons as well.

In order to achieve the better comparison results, we wanted to categorize the implementation process into 6 main categories such as

1. Defining dataset’s based on specific Domains and processing Datasets
2. Pre-Processing Interface to Process the Documents
3. Create Embedding’s for the input documents
4. Calculate Similarity using Cosine Similarity Algorithm based on programmatically generated embedding’s of input documents
5. Generate the output of Similarity Score as a CSV File
6. Utilizing the output data’s from CSV to generate meaningful results which shows semantic analysis between documents
7. **How did we create Dataset’s?**

During **t**he Initial research, we decided to make the analysis meaningful by classifying the datasets by domains; we come up the set of 5 to 10 words from each domain, **Example- Electricity and Energy** are the two different words by contextually it is related to the **Power Sector** domain and we wanted to ensure if this comparison using Open AI Embedding technique gives meaningful results by considering the contextual relevance, likewise we come up with more dataset around fifty to sixty different words from the same and different domain so that we could utilize such data for comparison and we also wanted to ensure it is editable by the developers to support changing the data for future analysis, Initially phrases comparison datasets are JSON formatted data which can be modified accordingly by the developers.

Document Comparison- We wanted our application to support comparison of any documents to show its contextual relevance between them, In order to do any comparison we need source documents and target documents, For Example, JobRequirment.txt is the source document and Job Profile A, Job Profile B is the target documents, so now when the source and target documents are compared, results will be meaningful and we could also come up the predefined threshold which can be managed by the application admin of any organization utilizing our application, Likewise we come up with few more meaningful document comparison by enabling the users of the app to add more documents either directly using file Manager or by enabling integration of user specific UI to our app to upload documents which requires development efforts, still we wanted to make sure application supports documents comparison dynamically making it more usable and researchable.

1. **Pre-Processing Interface to Process the Documents:**
2. **Create Embedding’s for the input documents:**

Interface CalculateEmbeddingAsync created for the purpose of accepting the phrases or documents as a text, and also additionally we want to ensure what is the category of the domain to find its relevance to the context it is created, hence it is designed to accept text 1, text 2 for processing source document or phrase in text1 and target document or target phrase in text2 and its corresponding filename respectively.

Task<double> CalculateEmbeddingAsync (string text1, string text2, string fileName1, string fileName2);

This Interface is made to utilize in the two different services called as SemanticSimilarityForDocumentsWithInputDataDynamic.cs and SemanticSimilarityPhrasesWithInputDataSet.cs where the actual implementation is created to produce the output embedding’s by utilizing the methods of open AI embedding nuget package.

OpenAIEmbeddingCollection collection = await client.GenerateEmbeddingsAsync (inputs);

GenerateEmbeddingsAsync () is the important method which accepts list of strings as inputs and produces collection of embedding as output for range of size 3052

Public static void PrintScalarValues (float [] embedding)

It is the custom method created to print the output of the each document or phrases as an embedding’s to print it individual values at every array till the end of the size of collection, Purpose of this method is utilize this values later during the visualization how closely or relatively the embedding’s at the vector space is being created. By knowing this difference it will help us to understand the correlation between the similarity score created VS Scalar values.

1. **Calculate Similarity using Cosine Similarity Algorithm based on programmatically generated embedding’s of input documents**

CalculateSimilarity (float [] embedding1, float [] embedding2);

This method is designed in such a way that it accepts the embedding’s generated from the previous GenerateEmbeddingsAsync (inputs) method, which will invoke the implementation of calculate Similarity by applying Cosine Similarity algorithm which is capable of returning the single similarity score as a double value as return type, values will be generated in the range between -1 to 1.

1. **Generate the output of Similarity Score as a CSV File**

Application is designed to support the generated output as a CSV file

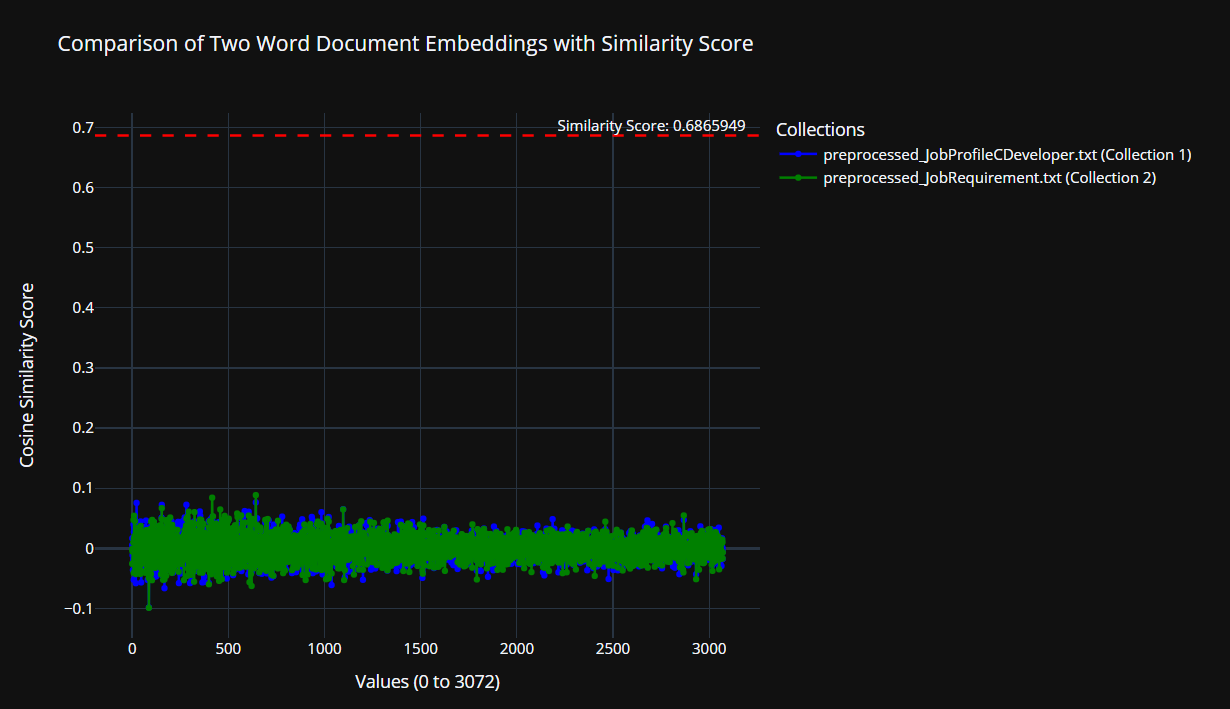
Public static void SaveResultsToCsv (List<DocumentSimilarity> results)

Public static void SaveResultsToCsv (List<PhraseSimilarity> results)

DocumentSimilarity and Phrase Similarity is the domain classes created to support the implementation of saving all the state of different values generated during the processing stage including the similarity score, domain, fileName1, fileName2 and score, domain, Phrase1, Phrase2, context respectively so that we achieve the clear understanding of what data’s can be mapped to which data to represent the generated data graphically using visualization methods.

1. **Utilizing the output data’s from CSV to generate meaningful results which shows semantic analysis between documents**

In order to meet the ultimate of understanding the Semantic Analysis of textual data between documents and create some meaningful output which would correlate to real time use cases**,** we used python as an external development tool to create a graphical chart, currently the application is designed to read the output CSV file generated dynamically if the files are placed in the root directory of the python app, so currently placement of output CSV required manual effort by the developers or the application admin,but we know this limitation which we are focusing during the later improvements either by us and paving way for other developers or ideas to improve the implementations. There are two types of plots we have designed; one chart is to graphically represent all the possible number of comparison of documents or phrases dataset designed by the user on X-axis VS its corresponding Similarity Score on Y axis; Other chart is designed in such way that developers are able to understand how the contextual relevance is actually generated by plotting its similarity score on Y axis VS Scalar Values (Ranges between 0-3052) on X-axis.



**Figure 3:**

**Scalar Values vs. SimilarityScorePlotForOneComparsion**

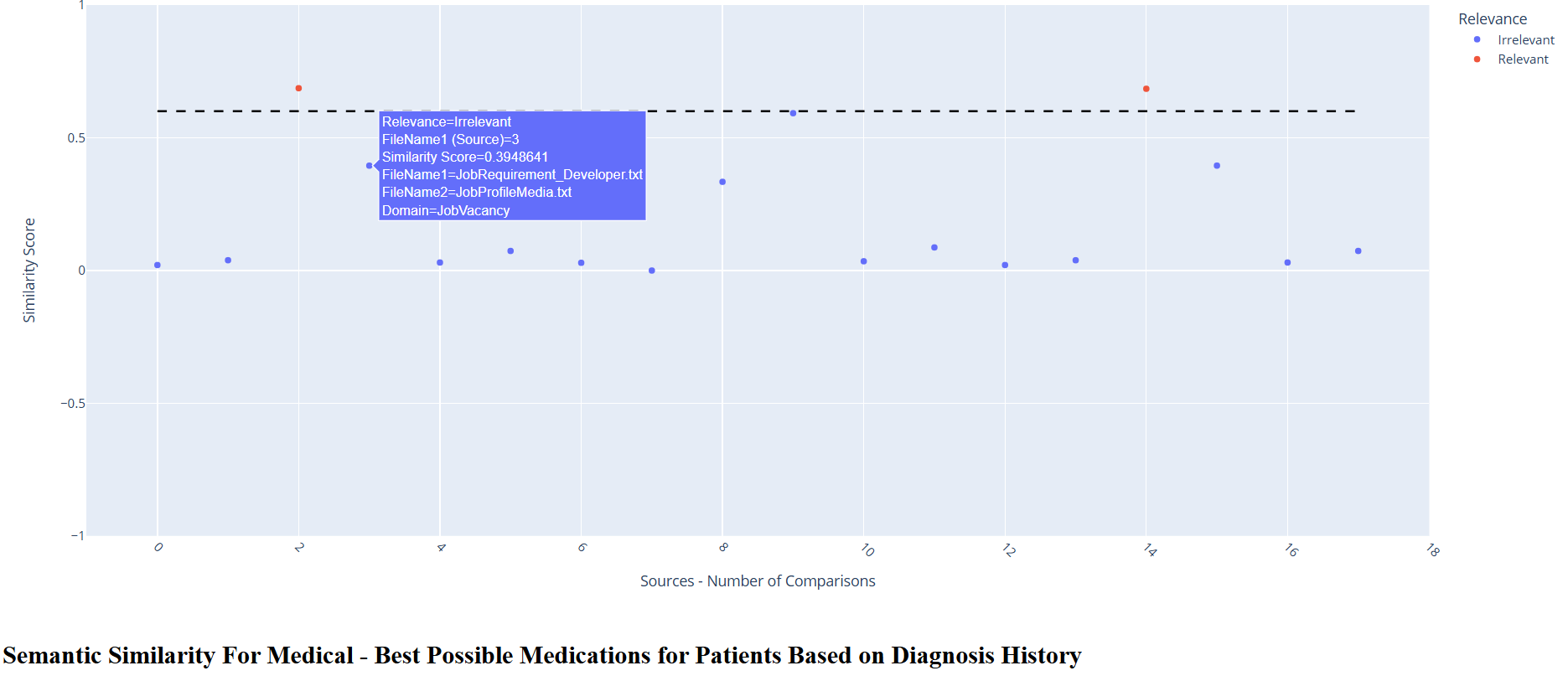


Fig 3: Semantic Similairty Similarity Score vs Number of Comparioson involved during Analysis

We wanted to display the analysis over a single chart as it gives an easy represenation of values