Strengthening Sentence Similarity Identification Through OpenAI Embeddings and Deep Learning

Dr. Nilesh B. Korade[1], Dr. Mahendra B. Salunke[2], Dr. Amol A. Bhosle[3],
Dr. Prashant B. Kumbharkar[4], Gayatri G. Asalkar[5], Rutuja G. Khedkar[6]

Assistant Professor, Department of Computer Engineering,
JSPM's Rajarshi Shahu College of Engineering, Tathawade, Pune-411033, Maharashtra, India[1, 6]
Assistant Professor, Department of Computer Engineering,
PCET's, Pimpri Chinchwad College of Engineering and Research, Ravet, Pune-412101, Maharashtra, India[2]
Associate Professor, Department of Computer Science and Engineering, School of Computing,
MIT Art, Design and Technology University, Loni Kalbhor, Pune-412201, Maharashtra, India[3]
Professor, Department of Computer Engineering,
JSPM's Rajarshi Shahu College of Engineering, Tathawade, Pune-411033, Maharashtra, India[4]
Research Scholar, Department of Computer Science and Engineering,
Shri Jagdishprasad Jhabarmal Tibrewala University, Vidyanagari, Churela-333001, Rajasthan, India[5]

***Abstract*— Discovering similarity between sentences can be beneficial to a variety of systems, including chatbots for customer support, educational platforms, e-commerce customer inquiries, and community forums or question-answering systems. One of the primary issues that online question-answering platforms and customer service chatbots have is the large number of duplicate inquiries that are placed on the platform. In addition to cluttering up the platform, these repetitive queries degrade the content's quality and make it harder for visitors to locate pertinent information. Therefore, it is necessary to automatically detect sentence similarity in order to improve the user experience and quickly match user expectations. The present study makes use of the Quora dataset to construct a framework for similarity discovery in sentence pairs. As part of our research, we have built additional attributes based on textual data for improving the accuracy of similarity prediction. The study investigates several vectorization methods and their influence on accuracy. To convert preprocess text input to a numerical vector, we implemented Word2Vec, FastText, Term Frequency-Inverse Document Frequency (TF-IDF), CountVectorizer (CV), and OpenAI embedding. In order to judge sentence similarity, the embedding offered by several approaches was used with various models, including cosine similarity, Random Forest (RF), AdaBoost, XGBoost, LSTM, and CNN. The result demonstrates that all algorithms trained on OpenAI embedding yield excellent outcomes. The OpenAI-created embedding offers excellent information to models trained on it and has significant potential for capturing sentence similarity.**

Keywords*—OpenAI; embedding; sentence similarity; FastText; Word2Vec; CNN; LSTM; precision; recall; F1-score

I. INTRODUCTION

The intricacy of natural language and the variety of ways in which phrases can express similar concepts make accurate sentence similarity assessment difficult. Scholars and professionals in the domain employ a variety of methodologies, which vary from conventional approaches such as cosine and Jaccard similarity to intelligent approaches that involve neural network models. An approach known as similarity identification or identical inquiry identification identifies similarities in the inquiries presented. The following, are several areas where text similarity matching is crucial to boosting service to clients [1].

- Client Assistance Chatbots Use similarity matching to find and group together related client inquiries. This increases the effectiveness of chatbot conversations and helps in generating consistent responses.
- By discovering and eliminating repetitive queries, community forums may enhance the user experience by making sure that conversations are concise and relevant [2].
- Improve customer service on e-commerce sites by recognizing common questions about the products and offering consistent replies.
- Employers can find comparable questions about policies, benefits, or procedures by using similarity matching in HR systems. This will help assure that responses are correct and consistent.
- To gain insights into popular topics and sentiment analysis, use similarity matching to combine and analyze similar questions or comments on social media networks [3].
- Similarity matching can be used by healthcare information systems to find related medical inquiries and give consumers reliable, consistent information about symptoms, diagnoses, and other health-related issues.

Finding such repetitively asked queries is crucial to improving the efficiency of resource utilization on the internet. It is not possible to find and remove duplicate questions manually. The duplicate inquiries or sentences should be identified automatically using some autodetection approaches [4,5,6]. In order to find similarities between two sentences, we

conducted our research using Quora's question-pair dataset. On the sentence dataset, we have used a variety of preprocessing approaches to eliminate unnecessary components and create a clean dataset that may be used for vectorization or embeddings. Based on parameters like sentence length, the frequency with which a string appears in the sentence, common strings in both sentences, fuzzy logic usage, etc., we have created a number of additional features that will provide additional information to a model trained on embedding. Numerous vectorization and embedding techniques, such as TF-IDF, CV, Word2Vec, FastText, and OpenAI text-embedding-ada-002 embedding, are used to convert text collections into numerical features. The metrics used to assess the quality of embedding and model performance on embedding include precision, recall, F1-score, and accuracy. The OpenAI text-embedding-ada-002 embedding shows potential in capturing sentence similarity and offers valuable information that supports different models for similarity identification.

The remainder of the document is structured as follows: Section II discusses the existing literature on duplicate question detection. Section III outlines the methodology, including the research flow, dataset, preprocessing steps, feature engineering, vectorization methods, and algorithm implementation. Section IV covers the evaluation of accuracy with various vectorization techniques and models. Section V presents a summary of our research findings and suggests avenues for further investigation.

II. Literature Survey

The sharing and learning environment have experienced significant changes due to the quick growth of digital platforms. Crowdsourced solutions like Community Question Answering (CQA) have been popular as a way for volunteers to share their knowledge and get their doubts regarding particular topics answered. A solution is required to address the issue of semantically comparable question detection for duplication identification in bilingually transliterated data. In order to detect question repetition, deep learning has been implemented by S. Rani et al. to evaluate informal languages like Hinglish, a bilingual blend of Hindi and English spoken on Community Question Answering (CQA) platforms. There are two components: the first is a language conversion component that creates a text in mono-language from input questions. The hybrid model, which combines a Siamese neural network (SNN), a capsule neural network, and a decision tree classifier, is used to determine the similarity between the question pairs. To calculate the similarity of questions, the SNN and the Manhattan distance function are utilized. An accuracy of 87% and an AUCROC value of 0.86 are obtained by validating the suggested model on 150 pairs of questions [7].

Contributors often make use of a pull-request procedure on social coding platforms like GitHub to present their source code modifications to inspectors of a particular repository. Due to the distributed nature of this approach, pull requests carrying out similar development tasks can be unintentionally submitted by multiple contributors, resulting in unnecessary effort and time spent reviewing. A strategy for allocating the same reviewer or reviewing team to each cluster of related pull requests was suggested by H. E. Salman et al., which makes it possible to save time and effort. To identify similarities across pull requests, first extract descriptive textual information from the content of the pull requests and use it to link equivalent pull requests together. To group relevant pull requests together, the K-means clustering and agglomeration hierarchical clustering algorithms are employed. The experimental results indicate that the K-Means algorithm achieves 94%, 91%, whereas agglomeration hierarchical clustering achieves 93%, 98% average precision and recall values over all evaluated repositories. The twenty popular repositories of public datasets are used to access the provided approach. In addition, the suggested method reduces the amount of time and effort required for reviews by using the K-Means algorithm by an average of 67% to 91% and the agglomeration hierarchical clustering technique by an average of 67% to 83% [8].

Millions of people use search engines every day to find solutions, which results in an increasing need for innovative, clever methods to assist people in solving problems. Using a 7GB real-time dataset, V. K. R. Anishaa et al. trained and evaluated four machine learning models in order to identify duplicate queries. The noise is eliminated by removing HTML tags, stop words, punctuation, white spaces, and URLs after the data has loaded. Pre-processing is carried out in SQLite databases utilizing PL/SQL blocks, which process enormous volumes of data faster than alternate techniques. The four different ML models are used to train the acquired dataset. After execution, the random, logistic regression, linear SVM, and XGBoost error parameters referred to from the log loss function are found to be, respectively, 0.887, 0.521, 0.654, and 0.357. As a result of the unique pre-processing activities carried out using PL/SQL, which improve response time overall, the result demonstrates that XGBoost is the best model, delivering the greatest accuracy in the shortest period of time [9].

On a social media platform where users post questions, other users can assist by editing the questions and providing more precise answers to the questions that are asked. Due to linguistic heterogeneity, it can be complicated to determine a sentence's true meaning with accuracy, making the classification of repeated inquiries a challenging process. Deep learning techniques have demonstrated exceptional performance in several natural language processing (NLP) problems, particularly in the area of semantic text similarity. In order to determine the semantic relevance between two queries, Z. Imtiaz et al. suggested a novel Siamese MaLSTM model, wherein the term "Siamese" refers to the employment of two or more sub-identical network architectures simultaneously and Ma indicates Manhattan distance. The GoogleNewsVector, FastText, and FastText subword word embeddings are used to independently train the Siamese LSTM model. The final prediction is then derived from the combination of these trained models [10].

Using a variety of techniques, many investigators have worked on duplicate text detection until now. Text data is pre-processed and converted to an array of numbers using the TF-IDF method. Using the Quora’s dataset, D. Basavesha examined five machine learning models. Adaboost yields an accuracy of 81.73%, random forest yields 81.72%, decision tree yields 79.29%, and logistic regression yields 79.21% [11].

A well-known software problem-solving website with a focus on solving errors in software code, Stack Overflow has seen an increase in visitors in recent years. L. Wang et al. employed Word2Vec to get the vector representations of words, and CNN, RNN, and LSTM are three distinct deep learning approaches that are taken into consideration to address the issue of similar inquiry discovery in Stack Overflow. The evaluation's findings demonstrate that WV-CNN and WV-LSTM have significantly outperformed the other baseline techniques. The dataset consisted of queries in various programming languages, including Perl, Java, and others. The outcome demonstrates that for every dataset, WV-CNN and WV-LSTM based on Word2Vec yield recall rates greater than 80% [12].

A. W. Qurashi measures the level of semantic equivalence across multi-word phrases for the regulations and guidelines stated in railway safety manuals. There are two text similarity measures that are examined: The cosine similarity metric maps the text into a vector space model, and the "Word2Vec" technique is used to determine the distance between the texts. A count-based metric called Jaccard similarity is the intersection of two sets divided by the union of two sets. The cosine similarity determines the degree of similarity between texts by converting sentences from documents into vectors using Word2Vec. The results show that the Jaccard similarity method, which measures similarity based on character matching, yielded unsatisfactory results and while evaluating the similarity of two documents, cosine provides a more accurate result by measuring the angle between vectorized phrases [13].

III. METHODOLOGY

A. Dataset

To determine textual similarity, this study makes use of the 0.4 million questions in the Quora dataset [14]. Table I presents information on the features and content of the dataset.

TABLE I. DATASET DETAILS

Attribute	Details
ID	Each dataset row is assigned a unique number that allows for its own unique recognition.
Question_ID1, Question_ID2	There is a distinct identity for each of the questions in the features labelled "Question No. 1" and "Question No. 2."
QuestionNo1, QuestionNo2	This feature includes real questions to check if they are similar to each other.
Is_Duplicate	When question pairs are intellectually examined, the result is Is_Duplicate, where 0 means false and 1 means yes.

B. Preprocessing

The dataset is subjected to basic preprocessing in order to be approved for usage in the succeeding phase, i.e., vectorization and model training. The database is examined for any duplicate or missing entries, and those that are found are discarded. All punctuation is deleted, the database is lowercased, and some special characters like '%' for percent are substituted out for their string equivalents. The URL and HTML elements were eliminated, and chartwords like "N.A." (which indicates "not applicable") were replaced. The dataset has been lemmatized after all stopwords have been eliminated and tokenization has been applied [15].

C. Feature Engineering

Strong features can increase the predictive ability of machine learning models by giving them pertinent information. On the basis of the properties of the question text that were listed in the dataset, new features were generated [16]. Table II presents an extensive overview of each newly developed feature along with a description.

TABLE II. FEATURE CREATION DETAILS

New Feature	Details
Sen1Len, Sen2Len	It is the entire sentence length, with all characters included.
WordCountSen1, WordCountSen2	Total number of words present in the sentence, including repeated words.
WordCommon	The number of terms that are present identically in the two sentences.
DistinctWords	It is a sum of unique words found in the first sentence and unique words found in the second sentence.
WordShare	It is the ratio of "word common" and "distinct words."
CWC_Min	It is a ratio of the number of common words *(WordCommon)* to the length of a shorter sentence *(min (Question1Length, Question2Length))*.
CWC_Max	It is a ratio of the number of common words *(WordCommon)* to the length of a larger sentence *(max (Question1Length, Question2Length))*.
CSC_Min	It is a ratio of the number of common stopwords (stopwords(sentence1) ∩ stopwords(sentence2)) to the minimum stopword count in the first and second sentences (min (stopwords count in sentence1, stopwords count in sentence2)).
CSC_Max	It is a ratio of the number of common stopwords *(stopwords(sentence1) ∩ stopwords(sentence2))* to the maximum stopword count in the first and second sentences *(max (stopwords count in sentence1, stopwords count in sentence2))*.
CTC_Min	It is a ratio of the number of common tokens *(tokens(sentence1) ∩ tokens (sentence2))* to the minimum token count in the first and second sentences *(min (tokens count in sentence1, tokens count in sentence2))*.
CTC_Max	It is a ratio of the number of common tokens *(tokens(sentence1) ∩ tokens (sentence2))* to the maximum token count in the first and second sentences *(max (tokens count in sentence1, tokens count in sentence2))*.
Avg_Tokens	It is a ratio of the sum of tokens present in both sentences to the number of sentences.
Abs_Len_Diff	It refers to the absolute value of the numerical difference between two sentence lengths.
Lng_Substr_Ratio	It is the ratio of the common longest substring in sentences 1 and 2 to the minimum token count from the first or second sentence *min (token count(sentence1), token count(sentence2))*.
Last_Word_Equal	The value is set to 1 if both sentences have identical last words; otherwise, it is 0.
First_Word_Equal	The value is set to 1 if both sentences have identical first words; otherwise, it is 0.

A Python library called fuzzywuzzy offers a variety of functions for string analysis and algorithm-based similarity score calculations [17]. The Levenshtein Distance (LD) is a measure of the degree of difference between two strings. The

degree of difference between the two strings increases with the number. is the smallest number of single-character modifications needed to convert one word to another. Let us assume that the lengths of the two sentences, sen1 and sen2, are l_1 and l_2, respectively. The LD, or the minimum number of modifications needed to transform sen1 into sen2, is D (l_1, l_2). By filling up a (l_1+1) × (l_2+1) matrix, the dynamic programming technique efficiently computes this distance.

$$\text{fuzz\_ratio} = \frac{1}{1+\text{LD}} \quad (1)$$

Using matching substrings of a given length, the fuzz_partial_ratio (FPR) function determines the "partial ratio" between two strings, which indicates how similar they are. The strings' higher fuzz_partial_ratio indicates a high degree of similarity.

$$\text{FPR} = \frac{2* \text{Common Characters in Two sentence}}{\text{len(sentence1)+len(sentence2)}} * 100 \quad (2)$$

The token_sort_ratio (TSortR) function, which computes the similarity ratio between two strings after sorting their tokens alphabetically, is especially helpful when working with strings that may contain the same words but in a different sequence.

$$\text{TSortR} = \frac{\text{LD}\left(\begin{matrix}\text{Sorted Tokens in Sentence1,}\\ \text{Sorted Tokens in Sentence2}\end{matrix}\right)}{\max\left(\begin{matrix}\text{len}\left(\text{Sorted Tokens in Sentence1}\right),\\ \text{len(Sorted Tokens in Sentence2)}\end{matrix}\right)} * 100 \quad (3)$$

The token_set_ratio (TSetR) function, which determines the similarity ratio between two strings based on the intersection and union of their unique tokens, is helpful when comparing texts that might contain common terms but also have differences.

$$\text{TSetR} = \frac{\text{LD}\left(\begin{matrix}\text{Token Set in sentence1,}\\ \text{Token Set in sentence2}\end{matrix}\right)}{\max\left(\begin{matrix}\text{len}\left(\text{Token Set in sentence1}\right),\\ \text{len(Token Set in sentence2)}\end{matrix}\right)} * 100 \quad (4)$$

D. Word Embedding / Vectorization

Word embedding is a method that captures syntactic and semantic similarities between words depending on their context of usage by representing words as vectors of real numbers in a high-dimensional space. The following vectorization and word embedding techniques were used in the study.

1) Count vectorizer: Count Vectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix [18]. The value of each cell is nothing but the count of the word in that particular text sample [19].

2) TF-IDF: The term frequency Inverse Document Frequency (TF-IDF) gives information about the more and less important words in a document. When retrieving information, a word's relevance within the text matters significantly. The more times a word appears in the text, the more significant it becomes. The frequency of a word (w) in a document (d) is measured by term frequency (TF), which is the ratio of a word's occurrence in a document to the total number of terms in the document [20]. A term's Inverse Document Frequency (IDF) indicates how common or uncommon a word is within the whole corpus of documents D. TFIDF is a product of TF and IDF, where the word that is more frequent in the document will get more importance and the word that is rare in the corpus will receive more weight [21, 22].

$$\text{Term Frequency}(\text{w}, \text{d}) = \frac{\text{number of times w appears in d.}}{\text{total number of words in document d.}} \quad (5)$$

$$\text{Inverse Document Frequency}(\text{w}, \text{D}) = \log \frac{\text{number of document in D.}}{\textit{number of documents contains w}} \quad (6)$$

$$\text{TFIDF}(\text{w}, \text{d}, \text{D}) = \text{TF}(\text{w}, \text{d}) * \text{IDF}(\text{w}, \text{D}) \quad (7)$$

3) Word2Vec: Words can be expressed as vectors using a technique called word embedding. Word embedding's main aim is to create low-dimensional feature vectors from the high-dimensional feature space of words while preserving contextual similarity within the corpus. Predicting the words that are close to each individual word in a sentence is the primary objective of the Word2Vec model. Word2Vec uses CBOW and skip-gram architecture, and using the training text input, it builds a vocabulary in order to the vector representation of words [23].

a) CBOW: A neural network termed the continuous bag-of-words (CBOW) model is used for NLP applications like text classification and language translation. One-hot encoding serves as the method for the target and input layer encoding, with a size of [1 X V]. The CBOW uses context words or surrounding words (X) as input in an attempt to predict the target or central word (Y). The weight sets (W, W') are initiated randomly, one between the input and hidden layers and the other between the hidden and output layers. Input Hidden layer matrix size is represented as [V X N], and hidden output layer matrix size is represented as [N X V], where N is a random number that indicates the size of our embedding space or how many dimensions, we want to use for describing our word. The hidden activation is the product of the input and the input-hidden weights. The product of hidden input and hidden output weights produces the output Y. The difference between the output and the target is evaluated and reported back to reset the weights [24,25].

b) Skip-Gram: Word2Vec also uses the skip-gram neural network approach, which reverses the CBOW architecture and predicts context or surrounding words (Y1, Y2…) for a given target word (X). The random weight is assigned after one-hot encoding of both the input layer and the target layer. To modify the weight assigned, the softmax function first calculates the probability of context words, then backpropagates the error by computing the loss between prediction and actual. Fig. 1 and 2 demonstrate the CBOW and skip-Gram structure [26,27].

4) FastText: An open-source platform called FastText, created by Facebook, enables professionals to acquire text representations and classifiers to perform efficient text classification. fastText offers subword embeddings by considering that words are made up of character n-grams [28].

For the purpose of removing common words, FastText generates a sample table. The theoretical foundation for this work is that words that are commonly used carry less information than uncommon terms and that a word's representation does not change much even after the same sentence is used several times [29]. In order to identify vector representations where the text and its associated labels have similar vectors, Fasttext represents text and labels as vectors. The softmax function is used to determine the the probability score of an accurate label given to its corresponding text [30].

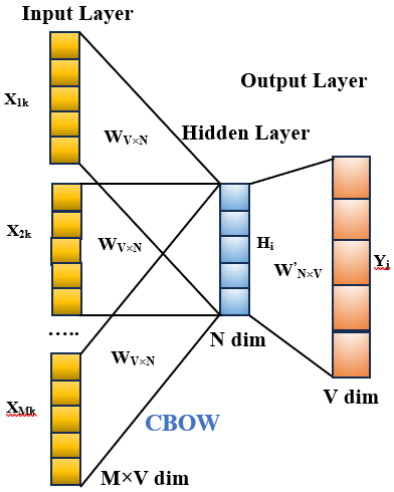

Fig. 1. CBOW architecture.

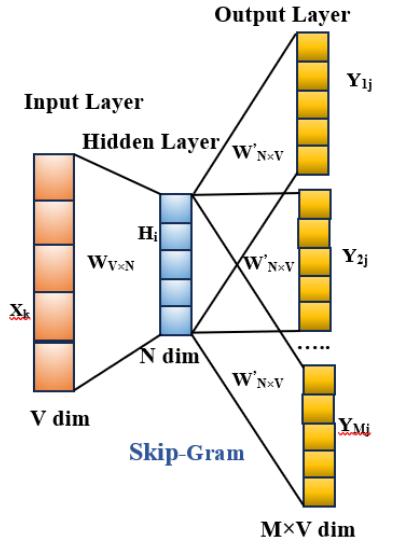

Fig. 2. Skip-gram architecture.

5) OpenAI: Text strings' relatedness can be evaluated by OpenAI's text embeddings. Searching, clustering, recommendations, anomaly detection, diversity assessment, and classification are among the common uses of OpenAI Embeddings [31]. The "text-embedding-3-small", "text-embedding-3-large", and "text-embedding-ada-002" are three robust third-generation embedding models offered by OpenAI. For text search, code search, and sentence similarity tests, text-embedding-ada-002 performs comparably to all previous embedding models, whereas for text classification, it achieves superior results [32]. The text-embedding-ada-002 context length is extended from 2048 to 8192 tokens by a factor of four, making it easier to work with lengthy documents. The new embeddings are only one-eighth the size of the davinci embeddings, with only 1536 dimensions and their maximum input token is 8191. Semantically comparable words are mapped to vectors that are close to each other in a continuous, dense, low-dimensional vector space. This is the basis for OpenAI embeddings, which use a sort of neural network called a transformer to represent text [33]. The conversion of text to vector by OpenAI embedding is explained in Fig. 3.

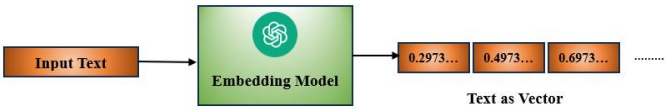

Fig. 3. OpenAI embedding.

E. Model Training

1) Cosine similarity: In natural language processing, cosine similarity is one of the metrics used to assess how similar two sentences are, regardless of their size. The cosine of the angle between two n-dimensional vectors projected in a multi-dimensional space is measured mathematically by the cosine similarity metric. A document's cosine similarity can range from 0 to 1, where 1 denotes that two vectors have the same orientation and 0 denotes that there is less similarity between the two documents. The following is the mathematical expression for the cosine similarity between two non-zero vectors [34,35].

$$Sentence\ Similarity = \cos(\theta) = \frac{P.Q}{||P|| * ||Q||} = \frac{\sum_{i=1}^{n} P_i Q_i}{\sqrt{\sum_{i=1}^{n} P_i^2}\sqrt{\sum_{i=1}^{n} Q_i^2}}\ where\ P\ and\ Q\ are\ vectors \quad (8)$$

2) Random Forest: The bagging approach is the foundation of the random forest ensemble learning method. Because decision trees have low bias and large variation, overfitting occurs if the tree grows too deep. By combining numerous decision tree predictions rather than relying just on one tree's output, Random Forest addresses this issue by reducing variance and resolving the overfitting issue. Decision trees are the foundation model used by Random Forest and predict the final output based on the majority of votes. When building a decision tree, entropy or the Gini Index is utilized as the splitting criterion [36].

$$Entropy = -\sum_{i=1}^{n} P_i log P_i\ \ where\ P_i\ is\ class\ label \quad (9)$$

$$Gini\ Index = \sum_{i=1}^{n} {p_i}^2\ \ where\ p_i\ is\ class\ label \quad (10)$$

3) AdaBoost: AdaBoost, or "adaptive boosting," is an ensemble machine-learning technique that builds decision stumps using decision trees. Adaboost combines weak learners into a single, powerful classifier to boost the performance of

machine learning algorithms. By assigning an equal weight to every data point, the adaboost algorithm initially constructs the model. In the subsequent iteration, the data points whose classification by the previous model was incorrect will be assigned greater weight [37, 38]. It will keep training models until it receives reduced errors. The following equation represents the final prediction.

$$Final\ Prediction = \alpha_1 * p_1 + \alpha_2 * p_2 + \cdots + \alpha_n * p_n \quad (11)$$

where, p1 represents the model's prediction and α1 represents the model's degree of significance [39].

4) XGBoost: A popular gradient boosting approach called XGBoost provides regularization that lets you control overfitting by applying L1/L2 penalties to each tree's weights and biases [40]. Following the base model's prediction, we build a decision tree using the residuals and splitting criteria, then determine the similarity scores of the root and leaf nodes using following equation.

$$Similarity\ Score = \frac{(\sum Residual)^2}{N+\lambda} \quad (12)$$

Where N is number of residuals and ƛ is regularization parameter. The following formula is used to compute the gain [41].

$$Gain = Left\ Similarity + Right\ Similarity - Root\ Similarity \quad (13)$$

In XGboost Regression, the Gamma Parameter Is Used. If the gain is less than the gamma value, the branch is cut, and no additional splitting occurs; otherwise, splitting proceeds. Pruning happens more often when gamma is higher. The XGboost Learning Rate is used to determine the model's convergence [42].

5) LSTM: One variation on a recurrent neural network that can identify and pick up on order dependence in sequence prediction issues is the Long Short-Term Memory (LSTM) network. The RNN cannot predict words that are held in long-term memory, but it can predict words based on recent data, which makes it unable to solve the long-term dependence problem. The LSTM is the type of neural network that receives an input (x_t) and outputs a value (h_t). The three gates in the memory are the input, forget, and output gates, which control information flow and have the capacity to add or remove data from the cell state, represented by the horizontal line at the top of the diagram. The forget gate has the responsibility for selecting which data should be erased from the cell state. The sigmoid layer generates a value between 0 and 1 after analysing h_{t-1} and x_t to determine which cell state data should be kept and which should be removed.

$$f_t = \sigma\big(W_f[h_{t-1}, X_t\,] + b_f\big) \quad (14)$$

The input gate is used to update the cell state value, taking into account both the previous time step's hidden state and the current input. The Sigma activation function in the first section determines the percentage of information that is needed. The Tanh activation function, which maps the data between -1 and 1, receives the two values in the second section. The input gate's output, which modifies the cell state determined by multiplying the outputs of the Tanh and Sigma functions [43, 44].

$$i_t = \sigma(W_i[h_{t-1}, X_t\,] + b_i) \quad (15)$$

$$C_t = \tanh(W_C[h_{t-1}, X_t\,] + b_C) \quad (16)$$

$$C_t = F_t * C_{t-1} + i_t * C_t \quad (17)$$

At the final stage, the sigmoid layer of the output gate decides which elements of the cell state are returned as output. The tanh layer modifies the cell's state to a value between 1 and -1, and the final output can be produced by multiplying the sigmoid layer's output by the tanh layer [45].

$$O_t = \sigma(W_o[h_{t-1}, X_t\,] + b_o) \quad (18)$$

$$h_t = O_t * \tanh(C_t) \quad (19)$$

6) CNN: CNNs are useful in NLP for linguistic modelling, autonomous translation, and classification of text. A variant of CNN known as 1D-CNN focuses on the analysis of one-dimensional data sequences, including text. By swiping the filter over the input matrix, the convolutional layers convolved the input, extracted features from the input, and passed the output to the next layer. CNN uses two parameters for regulating the size of an output matrix: stride, which indicates the number of pixels moved throughout the convolution process, and padding, which specifies the number of pixels added to an input matrix. These parameters control how the filter convolves across the input matrix. By multiplying the output matrix from the convolution layer and pooling matrix, the pooling layer tries to gradually reduce the spatial dimension of the representation in order to reduce the number of parameters and calculations in the network. The dropout layer attempts to minimize overfitting by randomly setting the input units to zero at each training phase. The feed-forward neural network uses the array as input for additional computation after it has been flattened into a one-dimensional array [46, 47].

7) Classification metrics: The true and false values accurately forecast are denoted by TP and TN. The false and true values that were incorrectly forecast are denoted by FP and FN. The ratio of exact forecasts to the total number of input observations is known as the classification accuracy. The ratio of correctly anticipated positive outcomes to all anticipated positive outcomes is known as precision, and the percentage of correct positive anticipations to all positive samples in the dataset is known as recall. The F1-score, which is the harmonic mean of recall and precision, is used when it's complicated to choose whether to go with recall or precision. Each metrics' corresponding mathematical equation is represented below [48, 49, 50].

$$\text{Accuracy} = \frac{[TP+TN]}{N} \quad (20)$$

$$\text{Precision} = \frac{TP}{[TP+FP]} \quad (21)$$

$$\text{Recall} = \frac{TP}{[TP+FN]} \quad (22)$$

$$\text{F1Score} = 2 * \frac{[\text{Precision} * \text{Recall}]}{[\text{Precision} + \text{Recall}]} \quad (23)$$

IV. Results and Discussion

The 30000 samples are chosen at random, and the bias in the dataset is controlled by making sure that each type of label has identical quantities in the dataset. The dataset was broken down into 6000 samples for testing and 24000 samples for training. Table III presents the classification evaluation metrics for the multiple techniques that were trained on CountVectorizer. The precision, recall, and f1-score were computed using a weighted average.

TABLE III. Performance of Count Vectorizer

Embedding	Model	Cosine	RF	AdaBoost	XGBoost	LSTM	CNN
CountVectorizer	**Accuracy**	0.565	0.781	0.745	0.792	0.771	0.790
	Precision	0.571	0.781	0.753	0.797	0.771	0.798
	Recall	0.565	0.781	0.745	0.792	0.771	0.790
	F1-Score	0.557	0.781	0.743	0.791	0.771	0.789
	FP	1716	843	1030	824	856	750

The outcome demonstrates that the Cosine Similarity calculated on CV yields disappointing results. In addition, in comparison to other models, the AdaBoost is producing unsatisfactory results. FP prediction refers to the situation where the model anticipated a negative value but the actual value was positive. When two statements have the same meaning but the model predicts they are different, it is normal; but, when two distinct sentences are anticipated to be similar, it is not acceptable and will negatively impact the system's performance. For this reason, we have taken into account FP, or Type-I mistake, as an evaluation metric.

The Table IV performance data for TF-IDF shows that the cosine similarity calculated on TF-IDF yields bad results, and the adaboost also yields disappointing results.

TABLE IV. Performance of TF-IDF

Embedding	Model	Cosine	RF	AdaBoost	XGBoost	LSTM	CNN
TF-IDF	**Accuracy**	0.660	0.778	0.756	0.779	0.767	0.791
	Precision	0.660	0.791	0.766	0.787	0.770	0.796
	Recall	0.660	0.778	0.756	0.779	0.767	0.791
	F1-Score	0.659	0.776	0.754	0.777	0.767	0.790
	FP	1099	972	1020	920	986	821

Table V performance data for Fasttext embedding demonstrates that all other techniques produce results that are reasonably nearby, whereas the cosine similarity calculated on Fasttext produces poor results.

Performance of each model trained using Word2Vec embedding is displayed in Table VI.

The performance of all models trained on OpenAI embedding is displayed in Table VII. The outcome highlights that, in contrast to alternative embeddings, cosine similarity yields outstanding outcomes. All models trained on OpenAI embedding show a performance gain of about 3%. With its ability to generate embeddings in such a way that two sentences with almost identical meanings have nearly the same value in the embedding, OpenAI has strong potential for capturing sematic meaning. With OpenAI embedding, the CNN performs effectively, yielding satisfactory outcomes with a slight FP value.

TABLE V. Performance of FastText

Embedding	Model	Cosine	RF	AdaBoost	XGBoost	LSTM	CNN
FastText	**Accuracy**	0.649	0.779	0.761	0.774	0.763	0.787
	Precision	0.651	0.784	0.763	0.780	0.763	0.790
	Recall	0.649	0.779	0.761	0.774	0.763	0.787
	F1-Score	0.648	0.778	0.760	0.773	0.762	0.786
	FP	1231	852	875	784	790	782

TABLE VI. Performance of Word2Vec

Embedding	Model	Cosine	RF	AdaBoost	XGBoost	LSTM	CNN
Word2Vec	**Accuracy**	0.671	0.767	0.767	0.782	0.776	0.791
	Precision	0.675	0.788	0.770	0.786	0.781	0.793
	Recall	0.671	0.767	0.767	0.782	0.776	0.791
	F1-Score	0.669	0.763	0.766	0.781	0.775	0.791
	FP	1219	982	854	831	871	762

TABLE VII. Performance of OpenAI

Embedding	Model	Cosine	RF	AdaBoost	XGBoost	LSTM	CNN
OpenAI	**Accuracy**	0.757	0.807	0.781	0.813	0.791	0.825
	Precision	0.768	0.812	0.781	0.817	0.796	0.823
	Recall	0.757	0.807	0.781	0.813	0.791	0.823
	F1-Score	0.755	0.806	0.781	0.812	0.790	0.823
	FP	1026	767	825	725	821	639

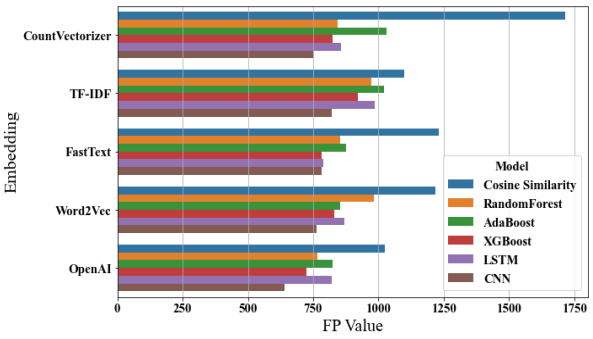

Fig. 4. Comparison of FP values for different models.

Fig. 4 indicates the false positive prediction for every model trained across different embeddings.

V. Conclusion

Detecting similarity across sentences can be helpful in the implementation of a number of different types of systems, such as question-answering systems, community forums, e-commerce consumer questions, instructional platforms, and chatbots for customer service. Sentence similarity must be automatically detected in order to meet user expectations promptly and enhance the user experience. Common uses for OpenAI embedded systems include searching, clustering, similarity, anomaly detection, diversity evaluation, and classification. The outcome demonstrates that machine learning receives valuable information from the embedding produced by the OpenAI model, improving prediction accuracy. Sentence similarity can be captured with significant potential using OpenAI embeddings. Almost all algorithms perform well with OpenAI embedding; CNN is one of the better performing algorithms. In order to improve prediction accuracy, we can incorporate a cascading CNN structure in future study. The CNN architecture's ideal parameter can be found using the optimization technique. Rather than taking a sentence as input, we can incorporate OpenAI STT and TTS to receive verbal input and produce verbal output in future research.

References

[1] N. B. Korade, M. B. Salunke, G. G. Asalkar, R. G. Khedkar, A. U. Bhosale, D. M. Joshi, and A. C. Jadhav, “Exploring NLP Techniques for Duplicate Question Detection to Maximizing Responses on Q&A Websites”, International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no.3, pp. 11-20, 2024.

[2] R. F. G. Silva, K. Paixão and M. de Almeida Maia, "Duplicate question detection in stack overflow: A reproducibility study," 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, Italy, pp. 572-581, 2018, doi: 10.1109/SANER.2018.8330262.

[3] J. Wang, and Y. Dong, “Measurement of Text Similarity: A Survey”, Information, vol. 11, no. 9, 2020, doi: 10.3390/info11090421.

[4] H. T. Le, D. T. Cao, T. H. Bui, L. T. Luong and H. Q. Nguyen, "Improve Quora Question Pair Dataset for Question Similarity Task," 2021 RIVF International Conference on Computing and Communication Technologies (RIVF), Hanoi, Vietnam, pp. 1-5, 2021, doi: 10.1109/RIVF51545.2021.9642071.

[5] A. Gupta, K. Sharma, and K. K. Goyal, “Computation of Similarity Between Two Pair of Sentence Using Word-Net”, International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 5s, pp. 458–467, 2023.

[6] X. Sun, Y. Meng, X. Ao, F. Wu, T. Zhang, J. Li, and C. Fan, “Sentence Similarity Based on Contexts”, Transactions of the Association for Computational Linguistics, vol. 10, pp. 573–588, 2022, doi: 10.1162/tacl_a_00477.

[7] S. Rani, A. Kumar, and N. Kumar, “Eliminating Data Duplication in CQA Platforms Using Deep Neural Model”, Hindawi Computational Intelligence and Neuroscience, vol. 2022, 2022, doi:10.1155/2022/2067449.

[8] H. E. Salman, Z. Alshara, A. D. Seriai, “Automatic Identification of Similar Pull-Requests in GitHub’s Repositories Using Machine Learning”, Information, vol. 13, no. 2, 2022, doi: 10.3390/info13020073.

[9] V. K. R. Anishaa, P. Sathvika, S. Rawat, “Identifying Similar Question Pairs Using Machine Learning Techniques”, Indian Journal of Science and Technology, vol. 14, no. 20, pp. 1635-1641, 2021, doi:10.17485/IJST/v14i20.312.

[10] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G. S. Choi and A. Mehmood, "Duplicate Questions Pair Detection Using Siamese MaLSTM," IEEE Access, vol. 8, pp. 21932-21942, 2020, doi: 10.1109/ACCESS.2020.2969041.

[11] D. Basavesha., and Y. S. Nijagunarya, Detecting Duplicate Questions in Community Based Websites Using Machine Learning, Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021, April 2021, doi:10.2139/ssrn.3835083.

[12] L. Wang, L. Zhang, and J. Jiang, “Duplicate Question Detection With Deep Learning in Stack Overflow”, IEEE Access, vol. 8, pp. 25964-25975, 2020, doi: 10.1109/ACCESS.2020.2968391.

[13] A. W. Qurashi, V. Holmes and A. P. Johnson, "Document Processing: Methods for Semantic Text Similarity Analysis," International Conference on Innovations in Intelligent Systems and Applications (INISTA), pp. 1-6, 2020, doi: 10.1109/INISTA49547.2020.9194665.

[14] Quora Question Pairs: https://www.kaggle.com/c/quora-question-pairs/data

[15] M. J. Wu, T. Y. Fu, Y. C. Chang and C. W. Lee, "A Study on Natural Language Processing Classified News," 2020 Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN), Rajpura, India, 2020, pp. 244-247, doi: 10.1109/Indo-TaiwanICAN48429.2020.9181355.

[16] N. Ansari, and R, Sharma, “Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study”, ACM Conference, 2020, doi: 10.48550/arXiv.2004.11694.

[17] Y. Du and H. Huo, "News Text Summarization Based on Multi-Feature and Fuzzy Logic," in IEEE Access, vol. 8, pp. 140261-140272, 2020, doi: 10.1109/ACCESS.2020.3007763.

[18] T. Turki, and S. S. Roy, “Novel Hate Speech Detection Using Word Cloud Visualization and Ensemble Learning Coupled with Count Vectorizer”, Applied Sciences, vol. 12, no. 13, 2022, doi: 10.3390/app12136611.

[19] R. Goyal, "Evaluation of rule-based, CountVectorizer, and Word2Vec machine learning models for tweet analysis to improve disaster relief," 2021 IEEE Global Humanitarian Technology Conference (GHTC), Seattle, WA, USA, pp. 16-19, 2021, doi: 10.1109/GHTC53159.2021.9612486.

[20] F. Lan, “Research on Text Similarity Measurement Hybrid Algorithm with Term Semantic Information and TF-IDF Method”, Advanced Pattern Recognition Systems for Multimedia Data, 2022, doi: 10.1155/2022/7923262.

[21] J. Ni, Y. Cai, G. Tang, Y. Xie, “Collaborative Filtering Recommendation Algorithm Based on TF-IDF and User Characteristics”. Applied Sciences, vol. 11, no. 20, 2021, doi:10.3390/app11209554.

[22] H. Vranken H, H. Alizadeh, “Detection of DGA-Generated Domain Names with TF-IDF”, Electronics, vol. 11, no. 3, 2022, doi:10.3390/electronics11030414.

[23] P. T. Hung, K. Yamanishi, “Word2vec Skip-Gram Dimensionality Selection via Sequential Normalized Maximum Likelihood”, Entropy, vol. 23, no. 8, 2021, doi: 10.3390/e23080997.

[24] X. Yang, K. Yang, T. Cui, M. Chen, L. He, "A Study of Text Vectorization Method Combining Topic Model and Transfer Learning", Processes, vol. 10, no. 2, 2022, doi: 10.3390/pr10020350.

[25] X. Xue, H. Wang, J. Zhang,Y. Huang, M. Li,and H. Zhu, "Matching Transportation Ontologies with Word2Vec and Alignment Extraction Algorithm", Journal of Advanced Transportation, 2021, doi: 10.1155/2021/4439861.

[26] Q. Du, N. Li, W. Liu, D. Sun,S. Yang,and F. Yue, "A Topic Recognition Method of News Text Based on Word Embedding Enhancement", Computational Intelligence and Neuroscience, 2022, doi: 10.1155/2022/4582480.

[27] R. Esmeli, M. Bader-El-Den and H. Abdullahi, "Using Word2Vec Recommendation for Improved Purchase Prediction," 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9206871.

[28] T. Yao, Z. Zhai and B. Gao, "Text Classification Model Based on fastText," 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), Dalian, China, 2020, pp. 154-157, doi: 10.1109/ICAIIS49377.2020.9194939.

[29] Fasttext official site: <https://fasttext.cc/docs/en/support.html>

[30] D. Jeon, J. Lee, J. M. Ahn, C. Lee, "Measuring the novelty of scientific publications: A fastText and local outlier factor approach", Journal of Informetrics, vol. 17, no. 4, 2023, doi: 10.1016/j.joi.2023.101450.

[31] E. J. Ciaccio, "Use of artificial intelligence in scientific paper writing", Informatics in Medicine Unlocked, vol. 41, 2023, doi: 10.1016/j.imu.2023.101253.

[32] OpenAI Documentation: https://platform.openai.com/docs/introduction

[33] K. I. Roumeliotis, N. D. Tselikas, "ChatGPT and Open-AI Models: A Preliminary Review", Future Internet, vol. 15, no. 6, 2023, doi:10.3390/fi15060192.

[34] I. L. Ansorena, "On the benchmarking of port performance. A cosine similarity approach", International Journal of Process Management and Benchmarking, vol.11, no.1, pp.101 – 114, 2021, doi: 10.1504/IJPMB.2021.112258.

[35] R.S. Ramya, Ganesh Singh, S. N. Sejal, K.R. Venugopal, S.S. Iyengar, L.M. Patnaik, "R2DCLT: retrieving relevant documents using cosine similarity and LDA in text mining", International Journal of Information and Communication Technology, vol.19, no.4, pp.391 – 422, 2021, doi: 10.1504/IJICT.2021.118576.

[36] M. Schonlau, and R. Y. Zou, "The random forest algorithm for statistical learning", The Stata Journal, vol. 20, no. 1, pp. 3-29, 2020, doi: 10.1177/1536867X20909688.

[37] C. Wang, S. Xu, J. Yang, "Adaboost Algorithm in Artificial Intelligence for Optimizing the IRI Prediction Accuracy of Asphalt Concrete Pavement", Sensors, vol. 21, no. 17, 2021, doi: 10.3390/s21175682.

[38] G. Sembina, "Building a Scoring Model Using the Adaboost Ensemble Model," 2022 International Conference on Smart Information Systems and Technologies (SIST), Nur-Sultan, Kazakhstan, pp. 1-6, 2022, doi: 10.1109/SIST54437.2022.9945713.

[39] D. Sudharson, S. Ashfia Fathima, P. S. Kailas, K. S. Thrisha Vaishnavi, S. Darshana and A. Bhuvaneshwaran, "Performance Evaluation of Improved Adaboost Framework in Randomized Phases Through Stumps," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, pp. 1-6, 2021, doi: 10.1109/ICAECA52838.2021.9675739.

[40] D. M. Alghazzawi, A. G. A. Alquraishee, S. K. Badri, S. H. Hasan, "ERF-XGB: Ensemble Random Forest-Based XG Boost for Accurate Prediction and Classification of E-Commerce Product Review", Sustainability, vol. 15, no. 9, 2023, doi: 10.3390/su15097076.

[41] D. A. -L. Mariadass, E. G. Moung, M. M. Sufian and A. Farzamnia, "Extreme Gradient Boosting (XGBoost) Regressor and Shapley Additive Explanation for Crop Yield Prediction in Agriculture," 2022 12th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, Islamic Republic of, pp. 219-224, 2022, doi: 10.1109/ICCKE57176.2022.9960069.

[42] T. R. Mahesh,V. Vinoth Kumar, V. Muthukumaran, H. K. Shashikala, B. Swapna, and Suresh Guluwadi, "Performance Analysis of XGBoost Ensemble Methods for Survivability with the Classification of Breast Cancer", Hindawi, Journal of Sensors, vol. 2022, , 2022, doi: 10.1155/2022/4649510.

[43] Z. Wang, S. Kim, I. Joe, "An Improved LSTM-Based Failure Classification Model for Financial Companies Using Natural Language Processing", Applied Sciences, vol. 13, no. 13, doi: 10.3390/app13137884.

[44] B. Nath Saha and A. Senapati, "Long Short Term Memory (LSTM) based Deep Learning for Sentiment Analysis of English and Spanish Data," 2020 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, pp. 442-446, 2020, doi: 10.1109/ComPE49325.2020.9200054.

[45] N. B. Korade, and M. Zuber, "Stock Price Forecasting using Convolutional Neural Networks and Optimization Techniques", vol. 13, no. 11, pp. 378-385, 2022, doi: 10.14569/IJACSA.2022.0131142.

[46] N. B. Korade, and M. Zuber, "Boost Stock Forecasting Accuracy Using The Modified Firefly Algorithm And Multichannel Convolutional Neural Network", Journal of Theoretical and Applied Information Technology, vol. 101, no. 7, pp. 2668- 2677, 2023.

[47] N. B. Korade, and M. Zuber, "Stock Forecasting Using Multichannel CNN and Firefly Algorithm", Proceedings of the 2nd International Conference on Cognitive and Intelligent Computing, pp. 447-458, 2023, doi: 10.1007/978-981-99-2742-5_46

[48] A. Gasparetto, M. Marcuzzo, A. Zangari, A. Albarelli, "A Survey on Text Classification Algorithms: From Text to Predictions", Information, vol. 13, no. 2, 2022,. doi: 10.3390/info13020083.

[49] Y. Liu, and S. Yang, "Application of Decision Tree-Based Classification Algorithm on Content Marketing", Journal of Mathematics, 2022, doi: 10.1155/2022/6469054.

[50] Y. I. Alzoubi, A. E. Topcu, A. E. Erkaya, "Machine Learning-Based Text Classification Comparison: Turkish Language Context", Applied Sciences, vol. 13, no. 16, 2023, doi: 10.3390/app13169428.