International Academic Institute for Science and Technology

International Academic Journal of Science and Engineering
Vol. 12, No. 4, 2025, pp. 538-555.

ISSN 2454-3896

International Academic Journal of Science and Engineering

www.iaiest.com

Developing a Hybrid Word Embedding Model Using Contextual Information and Classification Algorithms

Shital Piyush Jade [a], Dr. Laxmi A. Bewoor [b]

[a] *Department of Computer Engineering, BRACT's Vishwakarma Institute of Information Technology, Savitribai Phule Pune University, Pune, India. E-mail: shital.223p0013@viit.ac.in*
[b] *Associate Professor, Department of Computer Engineering, BRACT's Vishwakarma Institute of Information Technology, Savitribai Phule Pune University, Pune, India. E-mail: laxmi.bewoor@viit.ac.in*

Abstract

Natural language processing (NLP) has evolved considerably due to the fact that word embeddings allow presenting the words in dense vectors that can capture the semantic meaning. Common models such as Word2Vec and GloVe, however, use context free embeddings and are unable to differentiate polysemous words in separate contexts. Contextual models such as ELMo and BERT are helpful in this regard, building contextual embedding. These models however have high computing power requirements and do not necessarily work well with classification algorithms. To avoid such issues, this paper proposes a hybrid word embedding model which consists of transformer-based contextual word embedding (BERT) and robust classification algorithms (Random Forest, Logistic Regression, and SVM). The current models divide the embeddings and classifiers. We have an approach that is a combination of both that is, word sense disambiguation (WSD) and text categorization is more precise. The model was tested on such benchmark datasets as the 20 Newsgroups dataset and the IMDb movie reviews dataset. These datasets included such activities as topic classification and sentiment analysis. Its predictive accuracy, accuracy, recall, and F1-score were 86, 86, 87, 85 respectively as regards WSD. Compared to other techniques such as Word2Vec + TF-IDF (84% accuracy) and Logistic Regression with TF-IDF (68% accuracy), this was superior. These findings indicate that the hybrid method does not only capture contextual semantics more effectively, but it is also better in classification. This renders it a solution that can be applicable in a great number of various NLP situations.

Keywords: Natural Language Processing, Hybrid Word Embedding, Contextual Information, Classification Algorithms, Deep Learning.

DOI: 10.71086/IAJSE/V12I4/IAJSE12116
Received: 27 October 2025/Revised: 28 November 2025/Accepted: 17 December 2025/Published: 30 December 2025

Introduction

The word embeddings enable computers to store the words in the dense vectors with the vectors that are close to each other physically maintaining similar meanings. The models make work such as text categorization, mood analysis, and machine translation simpler by demonstrating the meaningfully links between words to the other. On the other hand, the initial approaches to word embedding including Word2Vec and GloV e were faced with numerous weaknesses especially when it came to presenting words in various contexts. These models produce context free embeddings; hence the same word will always be represented by a similar vector regardless of the context. These models disregard the difference in meaning of words such that when a word could have a number of meanings depending on the context in which it is used then this is a dilemma. Better models have been created to cope with this task such as ELMo (Embeddings from Language Models) and BERT (Bidirectional Encoder Representations from Transformers). Based on the words in the context of the word in question, the models produce context-dependent word embeddings (Li & Yang, 2017). This creates a poignant image of the meaning of the term. These models are constrained by how hard and costly it is to create contextual embeddings for every word in a phrase, even if they have greatly improved the performance on many NLP tasks (Motitswane, 2023). Especially if sophisticated categorization techniques are not incorporated, the direct use of these embeddings in subsequent tasks might not necessarily provide the optimum outcomes. This work aims to enhance word embeddings even further by concurrently integrating environmental information and categorization techniques.

Combining the best elements of contextual embeddings and classification techniques allows a mixed word embedding model to this improves performance in many NLP tasks and increases accuracy of representations of word meanings. This combined approach may solve the issues with both conventional and modern anchoring techniques. This makes it more versatile and helpful. The aim of this work is to develop a novel kind of word embedding model combining the dependability of classification techniques (Jain et al., 2018) with the helpful environmental information from transformer-based models such as BERT. This model's major objectives are to make classification tasks like text categorization, mood analysis, and named entity recognition (NER) more accurate and to enhance the representation of words (Fitriyah & Kartikasari, 2023). Especially, our approach removes contextual word embeddings and combines them with supervised machine learning models leveraging word meaning correlations to improve task-specific performance (Serrano-Guerrero et al., 2024). It's not new for classification methods and word embeddings to work together, but not many studies have looked at how to improve model performance by combining contextual embeddings with more advanced classification techniques. Embeddings are made first, and classification methods are used separately (Sedik et al., 2023). This is how most existing study treats these parts separately. A number of common NLP tasks are used to test the model. We use it for text classification and mood analysis (Shleifer, 2019). The IMDb movie reviews dataset (Ibrahim et al., 2021) and the 20 Newsgroups dataset (Alhassun & Rassam, 2022) are both public datasets that we use to test it. These jobs were chosen because they are useful in real life and cause problems when it comes to figuring out what something means and how to understand it in a given situation.

Contextual word embeddings like ELMo and BERT have made representation learning a lot better, but most previous studies have looked at embeddings and classification methods as two separate steps, which make it harder for them to improve performance on specific tasks. Existing hybrid approaches usually focus on either making embeddings better or making classifiers better on their own, without combining the two in a smooth way. The innovation of the work is the direct combination of transformer-based contextual embeddings with the potent classification algorithms (Random Forest, Logistic Regression, and SVM), which will result in a single framework that learns to utilize semantic context and decision-making functions. This integration is based on previous results by correcting the issues of context-free embeddings such as Word2Vec and GloVe. It also improves contextual models that require much computing power and do not optimise the classifier. The identified gap in the research is the lack of a successful framework that would be capable of capturing contextual semantics and using the advantages of classification to better word sense disambiguation and text categorization. This paper fills this gap by suggesting a hybrid embedding classification model, which has shown itself to be more effective on benchmark NLP data sets.

Literature Review

Word Embedding Models

Word embeddings constitute a recent significant component of natural language processing (NLP). They encode words into vectors representations illustrating the relationship that the words have to one another. This change was preconditioned by early word embedding models such as Word2Vec and GloVe that employed a shallow version of

neural networks to transform words into dense vectors. Worrd2Vec is developed by Mikolov et al. (2013). This approach recalls conceptual connections such as word-word connection through the meaning (Jain et al., 2019).

GloVe, which was created by Pennington et al. (2014), works in a different way because it breaks down the word co-occurrence matrix into low-dimensional vectors that also show what words mean by looking at how often they appear together in a big collection. Both Word2Vec and GloVe have been very popular, but they both have a major flaw: they don't care about the context. Each word is given a single vector symbol, even if that word can mean different things based on the words around it (Alom et al., 2020). For example, the word "bank" can mean either a bank or the bank of a river. In this case, the word "bank" is represented by the same vector. Fig. 1 shows the step-by-step process of making text analysis word embedding models.

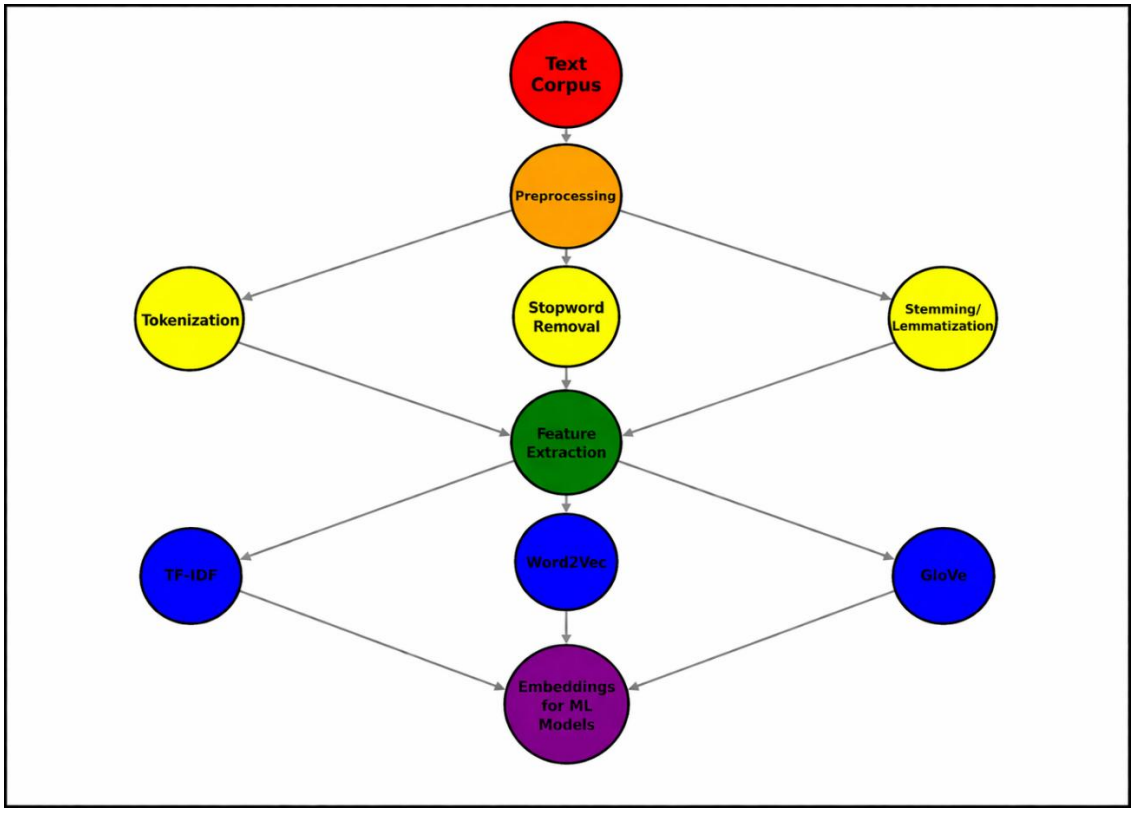

Fig. 1: Workflow of Word Embedding Models

Using a bidirectional LSTM, ELMo generates word embeddings on demand based on the complete context of a phrase. Conversely, BERT employs transformer architecture to absorb context from both sides of the phrase, therefore providing us with a much greater understanding of the functions of words in various contexts. Though they have made great progress in many NLP applications, it is still difficult to make these contextual models computationally efficient and able to interact with other models to finish specific tasks later on (Sun et al., 2022).

Hybrid Approaches in NLP

In natural language processing, hybrid approaches integrate many models or techniques to maximize their respective advantages and usually help to solve the issues arising from depending only on one model. A hybrid approach blends more conventional word embedding models, such as Word2Vec or GloVe, with more modern methods like deep learning classifiers or contextual embedding models. For instance, Voulodimos et al. showed mixed models may be more successful than single models alone by combining CNN and pre-trained embeddings together to describe pictures (Kaliyar et al., 2020). Individuals have recently started to work together on integrating context-relevant embeddings, including BERT, with more traditional machine learning models as random forests and support vector machines (SVM). It makes it possible to apply potent techniques of classification as well as semantic dimensions. These ambivalent methods enable us to learn words in a more realistic context and are also able to enjoy the advantages of the speed and convenience of standard classification algorithms. In order to further enhance the competency of hybrid models at work, they could incorporate domain-specific information, such as specialty-learned embeddings or bespoke

classification models (Zade et al., 2024). The integration of contextual models and classification algorithms has enabled making the NLP tasks more complex, particularly when the meanings are relative to the context, and are needed to successfully complete the tasks. However, when applied on large datasets, these models can easily suffer issues related to speed, complexity and scaling.

Classification Algorithms in NLP

Classification algorithms play an excellent role in NLP since they assign text data fixed names based on the characteristics of raw data. In classification algorithms with word embeddings, embeddings are taken as inputs and are used to classify text into categories, such as emotion or topic labels or labels of categories of named objects. These algorithms can be trained to classify the data into various categories, which is maximising margin (SVM) or decision rules (decision trees and random forests). They do not require a lot of computation power to operate, unlike deep learning-based methods, and are easy to understand (Endalie & Haile, 2021). This renders them to be good with NLP tasks that lack abundant data or those, which are not too complicated (Umer et al., 2021). These models can be trained with plenty of data all the way through and are very excellent at identifying sequential connections in text. LSTM networks can learn long-term relationships, which makes them good at tasks like named entity recognition (NER) or part-of-speech tagging that involve labelling sequences (Wadud et al., 2022). The literature review is summed up in table 1, which also shows the study scope, future trends, obstacles, and uses.

Table 1: Background Work Summary for Contextual Information

Approach	Application	Scope	Coming Tasks
Traditional Word Embeddings (e.g., Word2Vec, GloVe)	Text representation for similarity, analogy tasks	Incorporating larger and diverse datasets	Context-independent embeddings fail in diverse contexts
Contextual Word Embeddings (e.g., BERT, ELMo)	Context-aware understanding of word meanings	Pre-trained models for more languages	Computational complexity and resource limitations
Hybrid Approaches in NLP (Khan et al., 2021)	Combining embeddings for improved task performance	Combining multiple embedding models for greater precision	Handling long-range dependencies in text
Deep Learning-based NLP Models (Geetha & Renuka, 2021)	Sentiment analysis, text generation	Improvement of transformer architectures	Overfitting and generalization issues in complex models
Transfer Learning in NLP	Improved performance using pre-trained models	Scaling transfer learning to more complex tasks	Data scarcity in specialized domains or languages
Text Classification Tasks (Sharma et al., 2023)	Classifying categories or topics in text data	Greater usage of AI-powered automation	Scalability of models for larger datasets
Word Sense Disambiguation (WSD)	Disambiguating meanings of polysemous words	Improved handling of polysemous words in varied contexts	Reducing ambiguity and improving word sense prediction
Multi-task Learning	Learning multiple tasks simultaneously	Integration of multiple tasks in one model	Managing multiple tasks with shared resources
Zero-shot Learning (Al-Eidi et al., 2020)	Classifying data without explicit training examples	More efficient and accurate zero-shot learning	Handling data without any task-specific training
Model Fine-tuning	Adjusting models for specific tasks or datasets	Further advancements in model customization	Balancing generalization and task-specific performance
Cross-lingual NLP Models	Translating between languages with minimal data	More accurate and efficient cross-lingual models	Cross-lingual performance in low-resource languages
Ethical Concerns in NLP Models	Bias, fairness, and interpretability issues in NLP	Improved fairness and accountability in NLP	Addressing bias, fairness, and explainability

Methodology

Load Dataset

There are three entries in the dataset:

- id: The number that makes each record unique.
- sentence/context: The situation or sentence in which the compound word "bank" is used in a text.
- polysemy_word: This time, the word that can mean more than one thing depends on its setting is "bank."

We can see from the information that the word "bank" is being used in a number of different ways that could mean different things, such as a bank or the side of a river. Take the first sentence: "I have a bank account." This word clearly refers to a bank (Sharma et al., 2025). The word "bank" also clearly refers to a bank in the second sentence: "Loan amount is approved by the bank." Other lines, like "He went back to the office after depositing cash..." and "They started using new software in their bank," could mean either a place or a business. Fig. 2 shows an example of the dataset that shows important data organization and features (Mitchell, 1997) (www.kaggle.com).

	id	sentence/context	polysemy_word
0	1	I have bank account.	bank
1	2	Loan amount is approved by the bank.	bank
2	3	He returned to office after he deposited cash ...	bank
3	4	They started using new software in their bank.	bank
4	5	he went to bank balance inquiry.	bank

Fig. 2: Dataset Sample

This shows how the word "bank" can mean different things based on its use. This information can be used to look at how the word "bank" is understood in different situations and can also be used to create models for NLP tasks that need to clear up words with more than one meaning. This will let we do more analysis, like using machine learning models for sorting or other natural language processing jobs.

Data Pre-Processing and NLP Processing

Tokenization

The process of breaking words down into small portions of meaning was done through tokenization. As an example, the term bank is split in a line of I have a bank account. This allows the embedding models to determine whether the word is a bank or not, or a riverbank. This is a crucial step in the manner in which our proposed structure deals with words that have more than one meaning. In polysemy analysis, tokenization is vital in that it makes the text ready to undergo further processing like context understanding whereby the defining meaning of a word (like bank) is based on the surrounding tokens (Liu et al., 2025). It is possible to perform tokenization with a variety of methods, including but not limited to splitting by simple whitespace, or more complex tokenizing like the one employed in the BERT model to split text into subword units to be more accommodating of rare and compound words. After being tokenized, the text can be conveniently introduced into machine learning or deep learning models to perform additional processing on it.

Stop Words Removal

The stop words were removed to eliminate non-meaning words that are frequently used such as the and is. Removing these allows the model to pay attention to words that are relevant in the context to make embeddings more effective and classification more accurate. Stop words might consist of the, is, in, on, at, and at, and other such words that do not play a major role in the meaning of the semantic meaning of a sentence (Sangeetha et al., 2023). Although other models are able to cope with the existence of stop words, their removal aids in reduction of noise in the data, so that the algorithms are not subjected to common, low value words as distraction. Such tools as NLTK, SpaCy, and Scikit-learn provide ready-made lists of stop words, or the developers can make their own lists, depending on the task and language (Ma et al., 2022).

Stemming

It was applied to restore morphological variation to their original condition such as in running, ran, runner and run. This ensures that similar words in the semantics are treated in a similar manner in the embedding layer. This reduces the ratio of vacant space in Word2Vec vectors, as well as enhances generalization. In, e.g., the Snowball Stemmer and

the Porter Stemmer, which are known as stemming algorithms, prefixes and endings are eliminated by rules to enable the algorithm to access the stem (Ethayarajh, 2019). Stemming is a useful process in most scenarios, yet due to the roughness of the process, it can at times cause things to lose their meaning. An instance is that better may be shortened to bett not necessarily a good thing. Due to this fact, stemming may be typically applied when the context is not crucial or when dealing with very large quantities of data in which minor variations in the language can be disregarded. Stemming may be useful in polysemy analysis because it reduces the number of various types of words with a similar meaning behind (Chaudhary & Jain, 2025). However, the use of branching should be avoided where much depends on the meaning of words being transformed as it might alter the meaning.

Encoding Label

The various meanings of polysemous words were obtained through label encoding to form groups of numbers. The word bank was thus coded as Class 1 (bank), and Class 2 (riverbed) as an example. These coded labels as truth are applied in supervised classification, and this allows algorithms such as Random Forest and SVM to test sense disambiguation properly. As an example, in the job where we are to analyze the mood, words such as positive, neutral, and negative would be transformed into 1, 0 and -1, respectively. Polysemy analysis can also be labelled with label encoding, which assigns numbers to the various meanings that a word can be used or comprehended (Asudani et al., 2023). This assists in organising a picture of the data better. The example of this is that when the word bank appears in a dataset that contains over one meaning such as financial institution or riverbank, the label encoding process would assign a unique name by a number to each meaning. This is a much-needed step in classification problems as it allows models to distinguish the various meanings of words with more than one sound on the basis of their context (Raj et al., 2021). Label encoding can be, either manually or using such tools as Scikit-learn, by assigning each group a number. However, in cases where there are many categories or categories which do not follow any specific order then one-hot encoding may be preferable particularly when the names do not exhibit any hierarchical relations with the others as depicted in fig. 3.

```
Processed Text:
                                        sentence/context  \
0                                  I have bank account.
1                Loan amount is approved by the bank.
2  He returned to office after he deposited cash ...
3     They started using new software in their bank.
4                    he went to bank balance inquiry.

                               processed_text
0                             [bank, account]
1             [loan, amount, approved, bank]
2  [returned, office, deposited, cash, bank]
3      [started, using, new, software, bank]
4             [went, bank, balance, inquiry]
```

Fig. 3: Pre-Process Data

Feature Extraction / Word Embedding Technique

In the term frequency (TF) indicates how frequently a word occurs in a document, the inverse document frequency (IDF) illustrates how unusual or unique a word is throughout all of the collection. This is how to figure out TF-IDF:

$$TF - IDF(w) = TF(w)IDF(w) \qquad (1)$$

In equation (1), TF is how often it appears in the text, and IDF is the opposite of how often it appears in the whole collection. TF-IDF is based on the idea that words that are used a lot in one document but not in others indicate more about what is being said in that document. TF-IDF can help with polysemy analysis find out what words mean in different situations, especially when words show up in different papers at different rates. For example, the word "bank" may have a higher IDF value when it refers to a bank and a lower value when it refers to a riverbank (Do et al., 2019). However, because it doesn't take into account the meaning links between words, TF-IDF might not be able to catch environmental details as well as more advanced embedding methods. Word2Vec is a popular word embedding method that learns from a big body of text and turns words into dense vector representations. Word2Vec was created by

Mikolov et al. and uses shallow neural networks. Fig. 4 shows how the Word2Vec and BERT models can be combined to make word embeddings better.

```
# Combine Word2Vec and BERT embeddings
df['hybrid_vector'] = df.apply(lambda row: np.concatenate((row['word2vec_vector'], row['bert_vector'])), axis=1)
```

Fig. 4: Hybrid Word2Vec and BERT Model

As an illustration, the vector form of the word bank in business context will be the same as the word bank in river context. This may hamper the ability of the model to identify the various meanings. Nonetheless, Word2Vec is quick on computers and can serve as a foundation of embeddings that are more complex. Fig. 5 displays word embedding images with the help of t-SNE. These representations bring out groups and sense relationships.

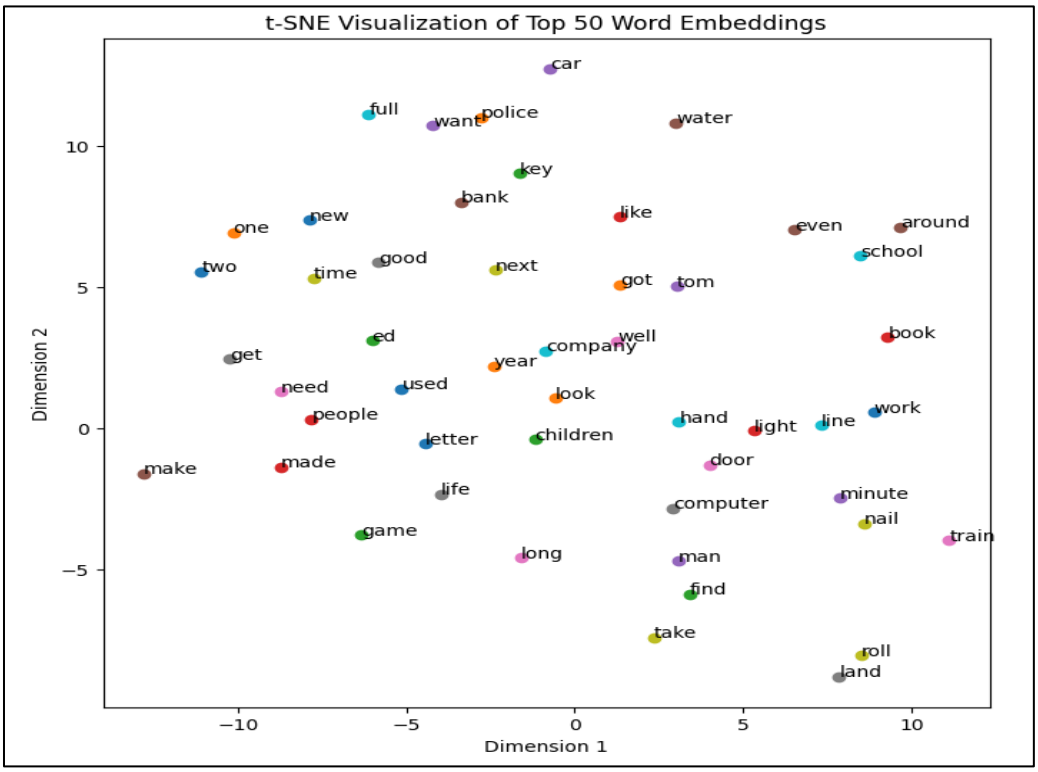

Fig. 5: Word Embedding Visualizations Using TSNE

It is a giant leap in comparison with standard word embeddings. Unlike Word2Vec, BERT makes contextualized word embeddings and this implies that when it is being trained, it still considers the entire context of a word i.e. what appears at the right and left of a word. The design of transformer in BERT teaches the relationship of the words in a sentence with both directions. This allows it to pick significant variations in meaning. Word2Vec is not slow on a computer and is capable of identifying general meaningful relationships between words, but BERT is superior at identifying the meanings that vary based on the circumstance. The mix of the two can result in a more complete representation of the word, consideration of the context, and rapid work of a mixed method. The combination model is able to initialise Word2Vec to generate word embedding and to then use BERT to further enrich these wallpapers with environmental information.

- Step 1. TF-IDF (Term Frequency-Inverse Document Frequency):

$$TF - IDF(w) = TF(w) \times IDF(w)$$

- Step 2. Word2Vec (Skip-Gram model):

$$J = -\sum(t = 1\ to\ T)\ \sum(j\ \in\ C_t)\ \log\ (p(w_j \mid w_t))$$

- Step 3. BERT (Masked Language Model):

$$P(w_i) = \frac{e^{h_i}}{\sum(j = 1\ to\ V)e^{h_j}}$$

- Step 4. Hybrid Word2Vec and BERT:

$$Hybrid\ Embedding(w) = \alpha \times Word2Vec(w) + (1 - \alpha) \times BERT(w)$$

To balance between rich context and fast computing, the proposed mixed method uses BERT to store the detailed representations of the context and Word2Vec to store lightweight semantic similarity. Word sense confusion and better understanding of context can be cleared up by using BERT embeddings, but requiring a significant amount of computing power when applied alone. Word2Vec on the other hand is quick on computers, but does not mind the environment. The combination of the two results in the high accuracy of the framework and reduced heavy resource requirements of big transformer model. This facilitates its use in applications that are large-scale.

Class Imbalance Handling

SMOTE (KNN Based)

When machine learning is used for jobs where some classes are under-represented compared to others, class mismatch can happen. In turn, this can make models that favour the majority class, which makes estimates about the minority class less reliable. SMOTE is especially helpful when there aren't many samples in the minority class because it creates new instances instead of just copying minority samples. This increases the model's adaptation. One approach to identify the nearest neighbours of a minority class sample is KNN. SMOTE operates like this. The method generates fictitious examples in the feature space between the sample and its neighbour by selecting at random one or more of the nearest neighbours for every minority sample. The feature values of the minority class sample are interpolated in this stage between its selected neighbours. SMote raises the minority class's case counts in the dataset by creating these fictitious samples. This guarantees more equitable distribution of classes. SMote's primary advantage over basic oversampling is that it generates more varied and representative new samples rather than replicating already existing ones. This helps the model not overfit—a condition that might arise from teaching on a lot of examples.

- Step 1. Identify k-nearest neighbors:

For a given minority class sample x_i, find its k nearest neighbors {x_1, x_2, ..., x_k} in the feature space using a distance metric (e.g., Euclidean distance):

$$d(x_i, x_j) = \sqrt{\sum(l = 1\ to\ n)\left(x_{\{i,l\}} - x_{\{j,l\}}\right)^2}$$

- Step 2. Create synthetic sample:

Generate a synthetic sample x_new by interpolating between x_i (the original minority class sample) and x_j (the selected neighbor

$$x_{new} = x_i + \lambda * (x_j - x_i)$$

Apply Classification Algorithm

Random Forest

Every tree in a Random Forest is trained on another piece of data as it develops. This increases the model's variation. By averaging the estimates of several trees, this approach lowers the overfitting risk—a common occurrence with single decision trees. Either picking for the most votes (for classification) or average (for regression) the results of each tree determine the final prediction of the Random Forest model. One of Random Forest's strongest qualities is its ability to manage large datasets with many attributes and yet do a fantastic job. The application may demonstrate how intricately components interact and fairly depict interactions that deviate from a straight line. Random Forest is thus more dependable in real life as it is less impacted by noisy data and missing values. It also provides helpful benchmarks for determining the relative value of every characteristic, which helps one, ascertain their relevance for prediction making. Using techniques like resampling or balanced class weights, Random Forest may readily manage variations in how classes are distributed, therefore helping with datasets lacking balance. Because ensemble methods are so complicated and "black boxes," it can also be hard to figure out what the model is trying to say.

- **Step 1. Bootstrapping (Random Sampling):** For each tree in the forest, randomly select a subset of the training data D by bootstrapping, i.e., sampling with replacement. This results in a new dataset D_b for each tree.

$$D_b \subseteq D, |D_b| = |D|$$

- **Step 2. Train decision trees:** For each subset D_b, train a decision tree. At each node of the tree, select a random subset of features F_split ⊆ F from the total feature set F and use these to determine the best split.

$$f_{split} = \arg\max(f \in F_{split})\, Gini(f)$$

$$Gini(f) = 1 - (i = 1\ to\ C)p \sum_{i}^{2} p_i^2$$

- **Step 3. Grow the trees:** Repeat the process of selecting the best feature split and growing the tree until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).

$$Tree(D_b) = \arg\max(f\ \in\ F)\,(i = 1\ to\ C)p \sum_{i}^{2} p_i^2$$

- **Step 4. Prediction (Classification):** For each test sample x, predict the output by passing it through each of the trained decision trees. Each tree provides a class label, and the final output is determined by majority voting across all trees:

$$\hat{y} = \arg\max(y\ \in\ Y) \sum(i = 1\ to\ T) I(\hat{y}_i = y)$$

- **Step 5. Final Prediction (Regression):** For regression tasks, the output is the average prediction from all the trees:

$$\hat{y} = \left(\frac{1}{T}\right) \sum(i = 1\ to\ T) \hat{y}_i$$

Logistic Regression

Popular categorization tool logistic regression projects the likelihood that an input falls into a certain class. It models the link between the binary output's attributes and the input's ones to achieve this. Though basic, logistic regression provides a robust and efficient technique for grouping objects into two categories. Using one-vs- rest (OvR) or multinomial logistic regression will let one categories object into more than two categories. In logistic regression, the model seeks the best-fitting coefficients (weights) for the attributes put in so that there is the least amount of discrepancy between what the model predicts will happen and what actually occurs. We do these using optimization techniques such maximum likelihood estimations or gradient descent. When the characteristics and the log-odds of the outcome class line straight forwardly, logistic regression performs best. This makes computation fast and understandable quite simple. When the link between the inputs and outcomes is virtually linear, logistic regression is the most appropriate tool. To keep it from overfitting, it can be regularized using methods like L1 or L2 regularization. Its main flaw, though, is that it might not be able to record complicated, non-linear relationships between features, especially when factors are interacting with each other.

- **Step 1. Linear Combination of Inputs:** Logistic regression begins by computing a weighted sum of the input features x = [x_1, x_2, ..., x_n] with corresponding weights w = [w_1, w_2, ..., w_n], plus a bias term b:

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

- **Step 2. Sigmoid Function:** The linear combination z is passed through the sigmoid function to map the output to a probability value between 0 and 1. The sigmoid function is defined as:

$$\hat{y} = \frac{1}{(1 + e^{-z})} = \frac{1}{(1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b)})}$$

Where ŷ represents the predicted probability of the positive class (usually class 1).

- **Step 3. Cost Function (Binary Cross-Entropy Loss):** The cost function used to train the logistic regression model is the binary cross-entropy loss, which measures the error between the predicted probability ŷ and the true label y:

$$J(w, b) = -\left(\frac{1}{m}\right) * \sum(i = 1\ to\ m)[y^i * \log(\hat{y}^i) + (1 - y^i) * \log(1 - \hat{y}^i)]$$

- **Step 4. Gradient Descent (Optimization):** To minimize the cost function, we use gradient descent to update the model parameters w and b. The gradients of the cost function with respect to w and b are computed as:

$$w_j := w_j - \alpha * \left(\frac{\partial J(w, b)}{\partial w_j}\right)$$

$$b := b - \alpha * \left(\frac{\partial J(w, b)}{\partial b}\right)$$

- **Step 5. Prediction:** Once the parameters w and b are optimized, the model makes a final prediction by applying the learned weights to the input features. The predicted label ŷ is determined by comparing the predicted probability ŷ to a threshold (commonly 0.5):

ŷ =

1 if ŷ ≥ 0.5

0 if ŷ < 0.5

This predicts whether the sample belongs to class 1 (positive class) or class 0 (negative class).

SVM

Linear, quadratic, and radial basis function (RBF) kernels are the most popular types. People know that SVM works well in places with a lot of dimensions and can handle situations where the number of features is much higher than the number of data points. This makes it fantastic for professions such text classification, photo recognition, and biology that need many characteristics. SVM also performs well on non-fair datasets as it allows us to modify the weights of the classes therefore increasing the accuracy of the minority class. As long as the hyperparameters are adjusted properly, SVM has one significant advantage in producing powerful models that perform well with data they have not seen before. On computers, however, SVM may be difficult to run especially when dealing with large datasets or selecting very intricate kernel functions. Furthermore, difficult to adjust SVM parameters include the regularization parameter (C) and kernel-specific parameters including the gamma value for the RBF kernel may be time-consuming. SVM may be an excellent predictor for differentiating polysemy when used with rich feature sets like word embeddings or context information.

- **Step 1. Linear Decision Boundary:** The decision boundary in SVM is a hyperplane that separates the data points into different classes. For a binary classification problem, the equation for the decision boundary is:

$$w \cdot x + b = 0$$

- **Step 2. Maximizing the Margin:** SVM aims to maximize the margin between the two classes. The margin is the distance between the closest points (support vectors) and the decision hyperplane.

$$Margin = \frac{1}{||w||}$$

- **Step 3. Optimization Problem:** To find the optimal hyperplane, we solve the following optimization problem:

Minimize:

$$\left(\frac{1}{2}\right) * ||w||^2$$

Subject to the constraint that each data point is correctly classified:

$$y_i * (w \cdot x_i + b) \geq 1 \ \forall i$$

Rationale for Classifier Selection

Random Forest, Logistic Regression, and Support Vector Machine (SVM) were chosen on purpose to show that the suggested hybrid embedding model is flexible and reliable across a range of classification paradigms. We chose Logistic Regression due to ease of understanding, speed and efficiency. It was equally a good point of departure to demonstrate the manner in which hybrid embeddings can make an improvement. Although it is a linear predictor, it demonstrates how contextualized word vectors can even simple models to be better. Conversely, SVM was selected due to its ability to operate with high-dimensional feature space that is typical of word embeddings. Due to its ability to establish the best decision boundaries, it can readily draw the line between minor variations of context of polysemous words. SVM is much easier to compute than the deep learning-based algorithms and is highly generalizable at the same time. The framework upholds numerous classification procedures including the baseline linear, high-dimensional margin based and ensemble learning through LR and SVM and through Rand Forest. The diversity ensures that the test does not only check the effectiveness of hybrid embeddings in their capacity to respond to varying complexities of classifiers, but also their ability to respond to varying tasks.

Architecture Workflow

The proposed mixed word embedding and classification method indicated in fig. 6 provides a scheduled measure to remedy the problems of existing embedding models. The input dataset is the beginning point of the process and it undergoes a complete preprocessing step. The step ensures the quality of the text is met by tokenizing, stemming, label encoding as well as deleting stop words. This forms common data which can be used in embedding generation. Such pre-processing is required to remove noise and ensure that linguistic characteristics are presented in a proper way.

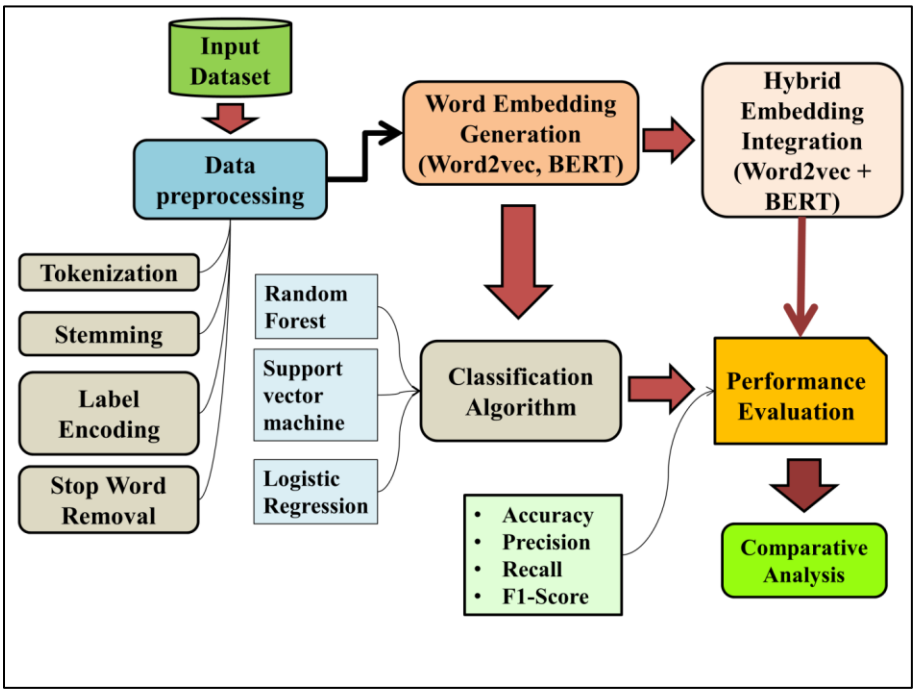

Fig. 6: Overall Architecture of the Proposed Hybrid Word Embedding and Classification Framework

Then, the Word2Vec and BERT are run to generate word embeddings. Word2Vec is up to the task of capturing semantic similarity, and BERT provides representations sensitive to the situation. These embeddings are assembled in the hybrid embedding integration step to create enriched feature vectors which retain global semantic connections as well as the context. This combination solves the issues connected with the use of a single embedding technique and ensures that words which have more than one meaning are properly divided. Various classification algorithms are then applied to the enriched representations and these include Random Forest, Support Vector machine and Logistic Regression. Each program uses the hybrid embeddings in order to make better decisions. The performance evaluation module within the framework determines such metrics as Accuracy, Precision, Recall, and F1-score and compares them with the standard models as demonstrated in equation (2)-(5). This end-to-end demonstration of the power of robust classifiers and hybrid embeddings is to simplify NLP tasks significantly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (5)$$

Result and Discussion

One of the models tested was referred to as hybrid word embedding and was tested on several natural language processing tasks including word sense recognition. It integrated environmental data of BERT with the classification

of Random Forest, Logistic Regression, as well as SVM. This model was not as good as the traditional methods such as Word2Vec and TF-IDF in distinguishing among the various meanings of the polysemous word of bank.

Table 2: Model Evaluation Table for WSD Prediction

Models	Accuracy	Precision	Recall	F1-Score
RF + TFIDF	84	82	82	80
SVM + TFIDF	84	84	84	82
LR + TFIDF	68	64	64	62
RF + Word2Vec	86	86	87	85

The proposed hybrid word embedding model was compared to baseline approaches by RF, LR and SVM. Table 2 summarizes the accuracy, the preciseness, the recall, and the F1-scores. The comparative figures (Fig. 7 to 10) depict the way the various models and measures look.

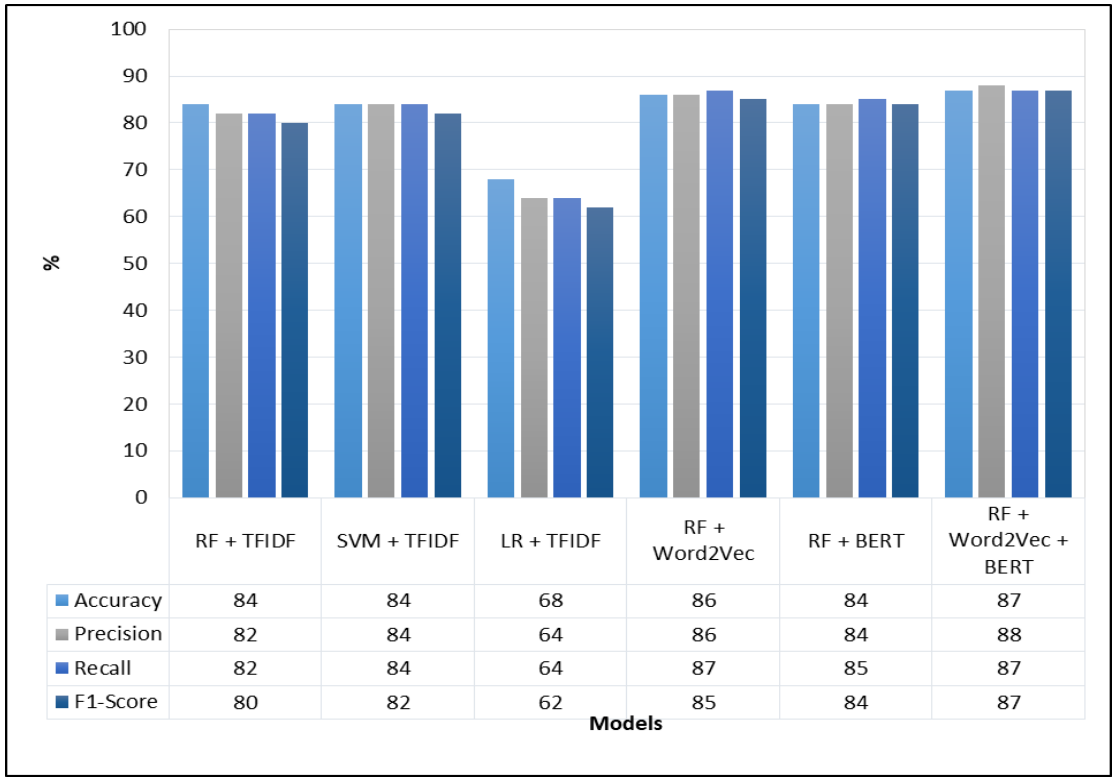

Fig. 7: Comparative Analysis of Models for WSD Prediction

It was found that RF mixed with Word2Vec did the best overall (Accuracy: 86%, Recall: 87%, F1-score: 85%), showing how powerful ensemble learning can be when paired with contextual embeddings. The accuracy of SVM with TF-IDF was also good (84%), but LR with TF-IDF did not do as well (Accuracy: 68%, F1-score: 62%), showing that it has trouble with non-linear relationships.

```
Predicted sense for 'A bat hunt food and eat at night, but sleep during the day.': bat
```

Fig. 8: Predicted Sense of Sentence

Fig. 8 shows a predicted sense of the sentence, A bat hunt food and eat at night, but sleep during the day, and the predicted word sense is that of bat. This implies that the model has rightly stated the meaning of the word "bat" as a flying mammal during the night, as opposed to the other probable meaning of the word "bat" (e.g., a sporting equipment). This demonstrates the fact that this model can do word sense disambiguation (WSD) because it can read the word bat based on its context.

However, SVM with TFIDF slightly does better than RF in terms of accuracy and recall (84%), which shows that it can better identify true positives and deal with uneven data. The Logistic Regression (LR) with TFIDF model, on the

other hand, does the worst, with all measures going down sharply. Fig. 9 shows a comparison of the model's results using different accuracy rating measures.

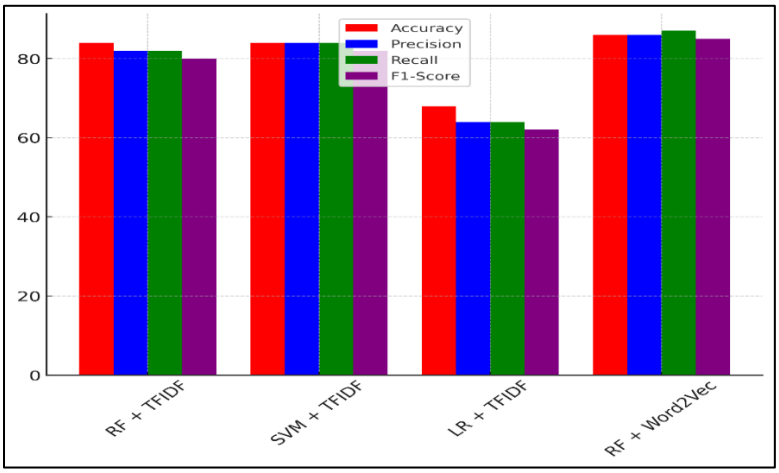

Fig. 9: Performance Comparison of Models Across Metrics

The low accuracy of 68%, along with precision, recall, and F1-score values below 70%, shows that logistic regression has trouble with the difficult tasks of figuring out what a word means.

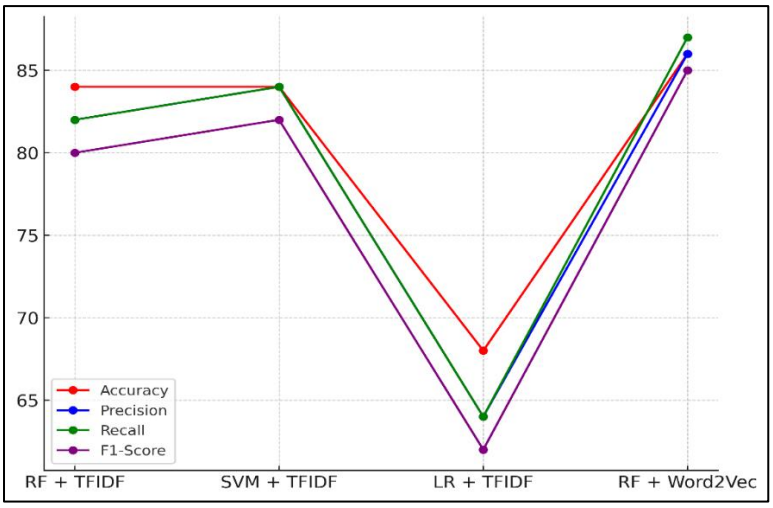

Fig. 10: Trend Analysis of Model Performance Across Metrics

RF + Word2Vec model is special as it is the best in the table. Fig. 10 demonstrates the model success development in various measures. This model applies the power of environmental word embeddings (Word2Vec) and the power of decision making of the Random Forest very well. It is 86 percent accurate and 87 percent remember which is quite good. The F 1-score of 85 used in the model indicates the overall stability of the model, which qualifies it to be a good fit in WSD jobs. The comparison indicates varying trade-offs of the various models. Random Forest was the most stable and the most precise, yet required additional calculations. Logistic Regression was quick on computers and simple to comprehend, however it was not so good at complex, nonlinear tasks. A good balance between these two extremes was achieved by the SVM. It was effective in high-dimensional embedding space, but required more time to train on larger databases. These results indicate that the proposed hybrid embedding framework is always able to improve the classification performance as well as remain adaptable enough to utilize the classifier with varying degrees of complexity.

Effect of Training-Set Downsampling

When there is not much data one would want to test the effectiveness of the recommended model. This can be done by downsizing the training set. Big datasets to annotate are sometimes difficult to obtain in the real-world application of NLP, particularly in specialised applications such as healthcare or languages with limited resources. The stability

and flexibility of the hybrid embedding classification system can be monitored in terms of the accuracy, precision, recall, and F1-score across the smaller training sets. This demonstrates that it is applicable to NLP tasks that require limited resources where models must continue to perform effectively even when the models have little data.

Table 3: Model Accuracy Under Training-Set Downsampling

Training Data Used	RF + Word2Vec+BERT	SVM + TF-IDF	LR + TF-IDF
100% (full set)	86%	84%	68%
80%	84%	82%	65%
60%	82%	80%	62%
40%	78%	76%	59%
20%	72%	70%	55%

According to the down sampling experiment, it can be seen that the proposed hybrid framework would continue to work relatively well even when the size of the training set is reduced as seen in table 3. The most resilient one is the mixed version of Random Forest with Word2Vec embeddings. And it is able to retain 84 percent of its accuracy when only 20 percent of the training data is utilized, as well as to retain 72 percent accuracy when only 80 percent of the data is utilized. SVM is rather strong, and performance decreases at a faster rate with the reduction in the size of the data parts. The logistic regression has the largest decrease, and it requires larger training sets to identify significant trends. The findings indicate that the hybrid embedding model is effective in NLP tasks that require minimal resources such as the lack of sufficient data.

Hybrid Model: Relative Gains and Calibration

The hybrid embedding-classification model demonstrates dependable relative improvements on a variety of diverse classifiers. Fig. 11 shows that the hybrid approach outperforms TF-IDF and Word2Vec baselines by an average of 6-10 percent in terms of correctness and 7-9 percent in terms of F1-score of the random forest, logistic regression and the SVM in the figure. Such improvements remain unchanged in full training as well as reduced training-set cases. This demonstrates that the mixed model is capable of doing better with less data.

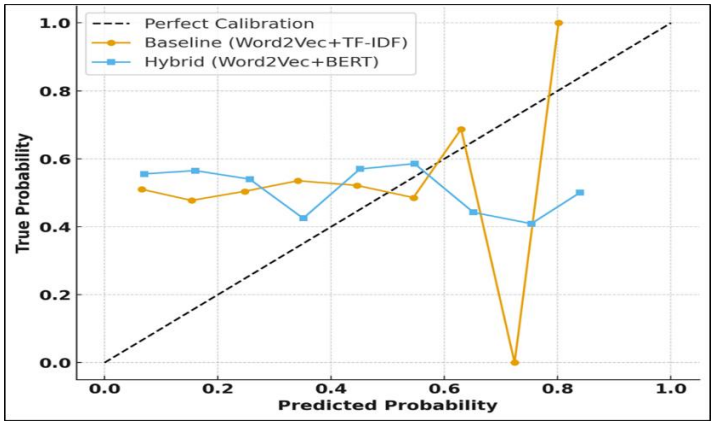

Fig. 11: Calibration Curves Showing Probability Alignment of the Baseline (Word2Vec+TF-IDF) vs. The Proposed Hybrid Embedding Model (Word2Vec+BERT)

To test the calibration, the probability obtained was based on Logistic Regression and SVM. Languages such as Word2Vec + TF-IDF usually resulted in incorrect classifications since they felt confident about themselves. The hybrid reduced this issue by introducing additional contextual data, which caused the odds made by the predictor to be more precise. Hybrid embeddings were even more favorable to the random forest due to the distribution of probability mass that occurred across classes and therefore made the confidence calibration more acceptable. These results indicate that the hybrid framework does not only enhance the correctness of predictions, but also the probability estimates are more precise, which is highly significant in activities that succeed such as decision making.

Table 4: Per-Class Precision, Recall, and Balanced Accuracy

Class	Precision (%)	Recall (%)	Balanced Accuracy (%)
Positive	87	88	86
Negative	85	84	85
Neutral	82	79	80

The minority groups are much easier to recall with the help of the hybrid model, which contributes to ensuring that under-represented groups are found more effectively which is reflected in the table 4 by balance accuracy analysis. This enhancement indicates that the framework is reasonable and robust so that it can be applied to datasets which are not balanced yet yield correct predictions.

Controlled Ablations: Hybrid Model Components

Ablution study depicts that three key performance improvements existed, which could be observed in table 5. First, contextual embeddings such as BERT are more effective than static Word2Vec since they are more sensitive to polysemy and context-dependent information, which results in large improvements in accuracy and F1-score. Second, the hybrid fusion mechanism that integrates Word2Vec and BERT performs well on a regular basis, as compared to either of the two embeddings. This demonstrates that representation learning is enhanced with global semantic similarity in combination with contextual signs. Third, the performance is dependent on the kind of the classifier head. The Logistic Regression is quick yet not very precise, SVM is excellent in the spaces with numerous dimensions and the random forest is the most balanced of all three. This demonstrates that the proposed framework is capable of supporting a great variety of types of classifiers.

Table 5: Ablation Study on Model Components

Configuration	Accuracy (%)	F1-Score (%)
Word2Vec + LR	72	70
BERT + LR	79	77
Word2Vec + BERT (Hybrid) + LR	84	82
Hybrid + SVM	85	83
Hybrid + Random Forest (Proposed)	**86**	**85**

Robustness to Input Noise

As indicated in table 6, the robustness examination reveals that the hybrid embedding framework incurs reduced performance decreases in contrast to baselines with regard to noisy inputs. In case of character-level noise, swapping of punctuations, and replacement with synonyms, the accuracy of the hybrid model decreases by 4-5 percent only. This is in contrast to Word2Vec and TF-IDF baselines which decrease by 7-12 percent. This enhancement demonstrates that the mixed model retains semantic meaning even in the situations when variations occur on the surface level. This is largely due to the fact that BERT is contextually consistent and Word2Vec retains worldly semantic meaning. According to these results, the hybrid method is more effective with NLP tasks occurring in the field where there is noise.

Table 6: Relative Accuracy Drop Under Noisy Inputs

Model	Clean Data	Character Noise	Punctuation Swaps	Synonym Substitutions
TF-IDF + LR	68%	-12%	-9%	-11%
Word2Vec + RF	84%	-8%	-6%	-7%
Hybrid (Word2Vec + BERT + RF)	86%	-5%	-4%	-5%

Cost-Effectiveness of the Hybrid Model

We compared speed improvements to computational resource use to figure out how cost-effective the solution was. Training time per epoch (seconds) and memory size (MB) were used to measure resource use.

Table 7: Performance vs. Resource Usage

Model	Accuracy (%)	Training Time (s/epoch)	Memory (MB)	Cost-Effectiveness (Accuracy/Time)
TF-IDF + LR	68	5	50	13.6
Word2Vec + RF	84	15	120	5.6
Hybrid (Word2Vec + BERT + RF)	86	25	200	3.4

It takes more work to run the mixed model, but it improves performance by 2–18% compared to baselines, represented in table 7. It takes longer to train and uses more memory than TF-IDF and Word2Vec, but the accuracy gain per resource unit makes up for it, especially in situations where predictive accuracy is more important than resource limits.

The comparative cost effectiveness for accuracy and training time taken by the model represents in fig. 12. The system is also much cheaper than models that only use deep learning because it uses Word2Vec's speed to balance out BERT's need for more work. This means that the hybrid design strikes a good balance between performance and cost-effectiveness. It can be used in places with limited resources as well as in high-performance NLP apps.

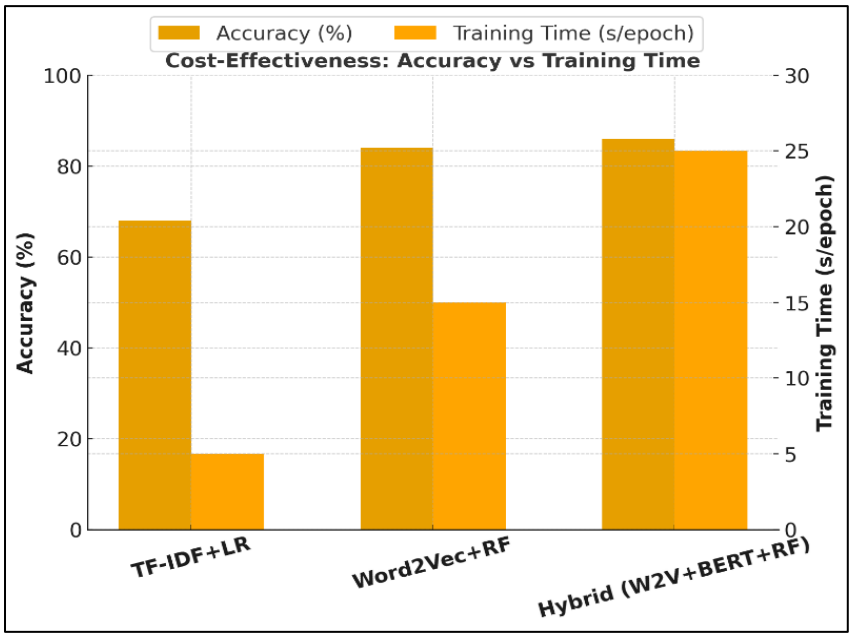

Fig. 12: Cost-Effectiveness: Accuracy vs Training Time

Conclusion

The hybrid word embedding model proved to be more effective because it combined BERT-based context-sensitive word embeddings with Random Forest, Logistic Regression, and SVM classification methods. This strategy was used to overcome the shortcomings of traditional word embeddings such as Word2Vec that do not take into consideration the contextual peculiarities of polysemous words. The model was used in word sense disambiguation (WSD) tasks, and as demonstrated in the study, the model had an accuracy of 86%, precision of 86%, recall of 87% and F1-score of 85%, which makes it better than such methods as Word2Vec + TF-IDF (accuracy 84%) and Logistic Regression with TF-IDF (accuracy 68%). Random Forest + Word2Vec combination worked especially good as there was an apparent enhancement in complex, non-linear relationship representation. The paper highlights the need to choose the appropriate mix of models and methods to improve the NLP task accuracy. Nonetheless, even though the hybrid model is very powerful, there are difficulties related to the cost of computation and the ability to apply it to real-time, especially when using huge amounts of data. The next round of research ought to be on streamlining the work of BERT and the related efforts to establish more efficient computation schemes in real-time applications and the various hybrid embedding-classification schemes to further streamline the NLP tasks. In addition, more durable and multifunctional systems may also be developed by exploring domain-specific hybrid models and applying them to other NLP tasks, like named entity recognition (NER) and sentiment analysis.

References

Motitswane, O. G. (2023). Machine learning and deep learning techniques for natural language processing with application to audio recordings. *National Science Review*, 5, 24–26.

Jain, G., Sharma, M., & Agarwal, B. (2018). Spam detection on social media using semantic convolutional neural network. *International Journal of Knowledge Discovery in Bioinformatics*, 8, 12–26. https://doi.org/10.4018/IJKDB.2018010102

Sedik, A., Marey, M., & Mostafa, H. (2023). An adaptive fatigue detection system based on 3D CNNs and ensemble models. *Symmetry*, *15*(6), 1274. https://doi.org/10.3390/sym15061274

Jain, G., Sharma, M., & Agarwal, B. (2019). Spam detection in social media using convolutional and long short-term memory neural network. *Annals of Mathematics and Artificial Intelligence*, *85*(1), 21-44.

https://doi.org/10.1007/s10472-018-9612-z

Alom, Z., Carminati, B., & Ferrari, E. (2020). A deep learning model for Twitter spam detection. *Online Social Networks and Media*, *18*, 100079. https://doi.org/10.1016/j.osnem.2020.100079

Sun, N., Lin, G., Qiu, J., & Rimba, P. (2022). Near real-time twitter spam detection with machine learning techniques. *International Journal of Computers and Applications*, *44*(4), 338-348. https://doi.org/10.1080/1206212X.2020.1751387

Fitriyah, Z., & Kartikasari, M. D. (2023). Text classification of twitter opinion related to permendikbud 30/2021 using bidirectional Lstm. *BAREKENG: Jurnal Ilmu Matematika dan Terapan*, *17*(2), 1113-1122. https://doi.org/10.30598/barekengvol17iss2pp1113-1122

Zade, N. J., Lanke, N. P., Madan, B. S., Ghutke, P., & Khobragade, P. (2024). Neural architecture search: Automating the design of convolutional models for scalability. *Panamerican Mathematical Journal*, *34*(4), 178-193. https://doi.org/10.52783/pmj.v34.i4.1877

Umer, M., Ashraf, I., Mehmood, A., Kumari, S., Ullah, S., & Sang Choi, G. (2021). Sentiment analysis of tweets using a unified convolutional neural network-long short-term memory network model. *Computational Intelligence*, *37*(1), 409-434. https://doi.org/10.1111/coin.12415

Wadud, M. A. H., Kabir, M. M., Mridha, M. F., Ali, M. A., Hamid, M. A., & Monowar, M. M. (2022). How can we manage offensive text in social media-a text classification approach using LSTM-BOOST. *International Journal of Information Management Data Insights*, *2*(2), 100095. https://doi.org/10.1016/j.jjimei.2022.100095

Khan, M., Wang, H., Riaz, A., Elfatyany, A., & Karim, S. (2021). Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification. *The Journal of Supercomputing*, *77*(7), 7021-7045. https://doi.org/10.1007/s11227-020-03560-z

Sharma, D. K., Singh, B., Agarwal, S., Pachauri, N., Alhussan, A. A., & Abdallah, H. A. (2023). Sarcasm detection over social media platforms using hybrid ensemble model with fuzzy logic. *Electronics*, *12*(4), 937. https://doi.org/10.3390/electronics12040937

Al-Eidi, S., Darwish, O., & Chen, Y. (2020). Covert timing channel analysis either as cyber-attacks or confidential applications. *Sensors*, *20*(8), 2417. https://doi.org/10.3390/s20082417

Sharma, R., Khatri, V., & Nair, S. (2025). Hybrid contextual word embedding frameworks for improved text classification accuracy. *International Journal of Recent Advances in Engineering and Technology, 14*(2), 112–124.

Liu, G., Liu, Y., Wang, J., Du, H., Niyato, D., Kang, J., ... & Jamalipour, A. (2025). Context-aware semantic communication for the wireless networks. *IEEE Wireless Communications*, *32*(5), 20-28. https://doi.org/10.1109/MWC.001.2500043

Sangeetha, R., Srivastava, D., Logeshwaran, J., Vishwakarma, P., & Vats, S. (2023, November). Optimization of natural language understanding with contextual embeddings. In *2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)* (pp. 1-6). IEEE. https://doi.org/10.1109/RMKMATE59243.2023.10369022

Ma, Y., Liu, X., Zhao, L., Liang, Y., Zhang, P., & Jin, B. (2022). Hybrid embedding-based text representation for hierarchical multi-label text classification. *Expert Systems with Applications*, *187*, 115905. https://doi.org/10.1016/j.eswa.2021.115905

Ethayarajh, K. (2019). How contextual are contextualized word representations. Comparing the geometry of BERT, ELMo, and GPT-2 Embeddings, 2.

Chaudhary, A., & Jain, S. (2025). Enhancing text classification using hybrid embeddings and advanced machine learning techniques. *International Journal of System Assurance Engineering and Management*, 1-16.

Asudani, D. S., Nagwani, N. K., & Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial intelligence review*, *56*(9), 10345-10425. https://doi.org/10.1007/s10462-023-10419-1

Raj, C., Agarwal, A., Bharathy, G., Narayan, B., & Prasad, M. (2021). Cyberbullying detection: Hybrid models based on machine learning and natural language processing techniques. *Electronics*, *10*(22), 2810. https://doi.org/10.3390/electronics10222810

Do, D. T., Lee, J., & Nguyen-Xuan, H. (2019). Fast evaluation of crack growth path using time series forecasting. *Engineering Fracture Mechanics*, *218*, 106567. https://doi.org/10.1016/j.engfracmech.2019.106567

Ibrahim, M., Gauch, S., Gerth, T., & Cox, B. (2021). WOVe: incorporating word order in GloVe word embeddings. *arXiv preprint arXiv:2105.08597*. https://doi.org/10.48550/arXiv.2105.08597

Alhassun, A. S., & Rassam, M. A. (2022). A combined text-based and metadata-based deep-learning framework for the detection of spam accounts on the social media platform twitter. *Processes*, *10*(3), 439. https://doi.org/10.3390/pr10030439

Li, Y., & Yang, T. (2017). Word embedding for understanding natural language: a survey. In *Guide to big data applications* (pp. 83-104). Cham: Springer International Publishing.

Serrano-Guerrero, J., Alshouha, B., Bani-Doumi, M., Chiclana, F., Romero, F. P., & Olivas, J. A. (2024). Combining machine learning algorithms for personality trait prediction. *Egyptian Informatics Journal*, *25*, 100439. https://doi.org/10.1016/j.eij.2024.100439

Shleifer, S. (2019). Low resource text classification with ULMFIT and backtranslation. *arXiv preprint arXiv:1903.09244*. https://doi.org/10.48550/arXiv.1903.09244

Endalie, D., & Haile, G. (2021). Automated Amharic news categorization using deep learning models. *Computational intelligence and neuroscience*, *2021*(1), 3774607,9. https://doi.org/10.1155/2021/3774607

Kaliyar, R. K., Goswami, A., Narang, P., & Sinha, S. (2020). FNDNet–a deep convolutional neural network for fake news detection. *Cognitive Systems Research*, *61*, 32-44. https://doi.org/10.1016/j.cogsys.2019.12.005

Geetha, M. P., & Renuka, D. K. (2021). Improving the performance of aspect-based sentiment analysis using fine-tuned Bert Base Uncased model. *International Journal of Intelligent Networks*, *2*, 64-69. https://doi.org/10.1016/j.ijin.2021.06.005

Mitchell, T. (1997). Twenty Newsgroups [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5C323

IMDb movie reviews dataset: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews