e-ISSN: 2614-8404
p-ISSN: 2776-3234

Hybrid Feature Combination of TF-IDF and BERT for Enhanced Information Retrieval Accuracy

Pajri Aprilio[1], Michael[2], Putu Surya Nugraha[3], Hasanul Fahmi[4]
[1,2,3]Faculty of Computer Science, Informatics, President University, Cikarang, Indonesia
[4]UNITAR International University, School of Information Technology, Selangor, Malaysia
Email: [1]pajri.aprilio@student.president.ac.id, [2]michael2024@student.president.ac.id,
[3]putu.nugraha@student.president.ac.id, [4]fahmi.zuhri@unitar.my

Abstract – Text representation is a critical component in Natural Language Processing tasks such as information retrieval and text classification. Traditional approaches like Term Frequency-Inverse Document Frequency (TF-IDF) provide a simple and efficient way to represent term importance but lack the ability to capture semantic meaning. On the other hand, deep learning models such as Bidirectional Encoder Representations from Transformers (BERT) produce context-aware embeddings that enhance semantic understanding but may overlook exact term relevance. This study proposes a hybrid approach that combines TF-IDF and BERT through a weighted feature-level fusion strategy. The TF-IDF vectors are reduced in dimension using Truncated Singular Value Decomposition and aligned with BERT vectors. The combined representation is used to train a fully connected neural network for binary classification of document relevance. The model was evaluated using the CISI benchmark dataset and compared with standalone TF-IDF and BERT models. Experimental results show that the hybrid model achieved a training accuracy of 97.43 percent and the highest test accuracy of 80.02 percent, outperforming individual methods. These findings confirm that combining lexical and contextual features can enhance classification accuracy and generalization. This approach provides a more robust solution for improving real-world information retrieval systems where both term specificity and contextual relevance are important.

Keywords – TF-IDF, BERT, Text Classification, Information Retrieval, Hybrid Model, Semantic Embedding, Neural Network

I. INTRODUCTION

Natural Language Processing (NLP) has an important role in enabling machines to interpret and process human language for tasks such as information retrieval and text classification. Accurate text representation is fundamental to the success of these tasks. Traditional methods like Term Frequency-Inverse Document Frequency (TF-IDF) have long been used due to their simplicity and effectiveness in representing term importance [1]. However, these methods are limited in capturing the semantic and contextual meaning of language, leading to performance issues in complex NLP tasks. On the other hand, deep learning models such as Bidirectional Encoder Representations from Transformers (BERT) have demonstrated a better performance by understanding the contextual relationships between words in a sequence, thus enabling better text understanding.

TF-IDF has been widely adopted in IR systems due to its effectiveness in term-level statistical analysis [2]. Despite its popularity, its inability to understand context within a document limits its performance on complex linguistic tasks. To address this, models like BERT utilize deep bidirectional transformers to learn contextual representations of words and sentences, significantly enhancing performance across various NLP benchmarks including question answering and document ranking [3][4].

Recent developments in IR have leveraged dense representations from models like BERT for semantic search and document matching [5][6]. Studies such as [7] demonstrate that integrating contextual embeddings into ranking systems can significantly boost relevance modeling. Furthermore, Nogueira et al. show how BERT can improve re-ranking effectiveness in passage retrieval tasks [8]. Yang et al. present simple yet effective applications of BERT in ad hoc retrieval, reinforcing its adaptability to diverse IR tasks [9].

However, while BERT excels at capturing the contextual meaning of words, it may underemphasize the importance of exact term matches which is a strength of traditional statistical approaches like TF-IDF. To bridge this gap, hybrid models combining statistical and contextual features are developed. Lin et al. highlight the potential of integrating pre-trained transformers with traditional signals to enhance ranking performance [10]. Moreover, the BEIR benchmark, developed by Thakur et al., supports evaluating hybrid models across heterogeneous datasets and demonstrates the effectiveness of combining retrieval paradigms [11].

Recent efforts in dense retrieval models have focused on pretraining and optimizing contextual embeddings for semantic matching. Gao and Callan [12][14] proposed corpus-aware language model pretraining, showing that language models tailored to the domain can significantly improve dense retrieval results without requiring extensive supervised data. Likewise, Zhan et al. [15] highlighted the importance of hard negatives in training dense models, enabling them to learn more effective decision boundaries for relevance. These advances validate the importance of fine-tuning semantic embeddings, but they still underutilize lexical signals like those found in TF-IDF.

Hybrid retrieval models remain relatively underexplored in comparison to dense-only methods, even though fusion strategies may offer improved robustness. For example, Qu et al. [16] introduced RocketQA, an optimized training framework for dense passage retrieval, but did not integrate traditional IR features like TF-IDF. Xiong et al. [17] proposed an efficient contrastive learning method, yet similarly focused solely on neural representations. In contrast, the model proposed in this study bridges statistical and semantic paradigms through a feature-level fusion strategy that balances term specificity with contextual understanding.

Furthermore, emerging techniques such as multiview and topic-aware training have demonstrated promise in improving generalization across domains. Ma et al. [18] introduced a multi-granularity view approach to dense retrieval, while Hofstätter et al. [19] presented a sampling strategy that enhances topic coverage during training. These approaches suggest that hybrid and diversified representation schemes can better address real-world IR scenarios. By combining TF-IDF and BERT features in a weighted manner, this study contributes to filling the gap in fusion-based IR models and aligns with current efforts to improve retrieval robustness.

Despite the promise of hybrid approaches, many existing works focus on dense representations alone or offer limited insights into effective fusion strategies between statistical and contextual features. Furthermore, evaluations are often conducted on limited datasets, which restrict the understanding of their generalization capabilities in real-world applications.

This study proposes a novel hybrid model that integrates TF-IDF statistical features with BERT semantic embeddings via a weighted feature-level fusion strategy. By reducing TF-IDF vector dimensions and aligning them with BERT embeddings, the proposed method creates a unified vector representation that captures both explicit keyword importance and deep contextual semantics. This dual-layer representation is then fed into a fully connected neural network for binary classification in an information retrieval context. The model is evaluated on the CISI benchmark dataset and demonstrates improved classification accuracy compared to standalone TF-IDF and BERT models.

The main objective of this study is to develop and evaluate a hybrid text representation model that combines TF-IDF and BERT for improved performance in information retrieval tasks. The goal is to leverage the complementary strengths of statistical and contextual representations to enhance relevance prediction accuracy.

II. RESEARCH METHODOLOGY

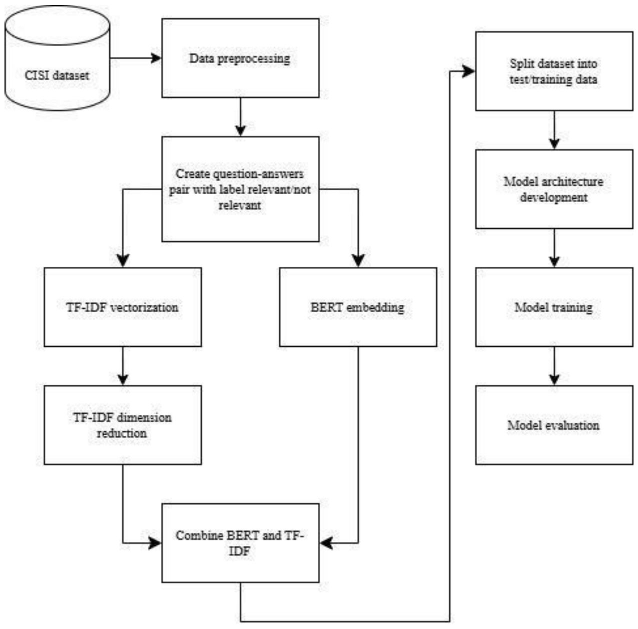

CISI dataset
Data preprocessing
Create question-answers pair with label relevant/not relevant
TF-IDF vectorization
BERT embedding
TF-IDF dimension reduction
Combine BERT and TF-IDF
Split dataset into test/training data
Model architecture development
Model training
Model evaluation

Fig. 1 Research Flow Diagram for TF-IDF + BERT Combination Methodology.

Figure 1 shows the flow of this research. This research used the CISI dataset [20], which contains classic query-document pairs for information retrieval tasks, it is a collection of information science research articles and queries commonly used for evaluating information retrieval and document ranking systems (https://ir.dcs.gla.ac.uk/resources/test_collections/cisi/). The research involves several steps which includes Data preprocessing, question-answer pair generation, TF-IDF vectorization and dimension reduction, BERT embedding, combining BERT and TF-IDF, dataset splitting, model architecture development, model training, and model evaluation. The methodological steps are outlined as follows.

A. Dataset Description

The experiment in this study uses the CISI dataset, a well-known collection in the field of information retrieval. The dataset consists of scientific document abstracts and user queries. The dataset contains three main files: CISI.ALL, CISI,QRY, and CISI.REL. CISI.ALL contains 1460 document abstracts, each with a unique ID and content. Table 1 shows the sample of document content. CISI.QRY contains 112 user queries related to the documents. Table 2 shows a sample of the user query. CISI.REL contains 3114 relevance data that maps each query to its corresponding relevant documents. Table 3 shows the sample of relevance judgement documents.

Table 1. Sample content of CISI.ALL file

Document Id	Document
1	18 Editions of the Dewey Decimal Classifications Comaromi, J.P. The present study is a history of...
2	Use Made of Technical Libraries Slater, M. This report is an analysis of 6300 acts of use in 104...
3	Two Kinds of Power An Essay on Bibliographic Control Wilson, P. The relationships between the...
4	Systems Analysis of a University Library; final report and research project Buckland, M.K. The...
5	A Library Management Game: a report on a research project Brophy, P. Although the use of games in pr...

Table 1 presents a sample of the content from the CISI.ALL file, which is a component of the CISI test collection. The table includes severa entries, each comprising a unique Document ID and the corresponding text excerpt from the document. These excerpts illustrate the variety and structure of bibliographic content found in the dataset, which typically includes titles, author names, and partial abstracts. The dataset is valuable for evaluating information retrieval algorithms due to its structured and semantically rich textual data.

Table 2. Sample content of CISI.QRY file

Query Id	Query
1	What problems and concerns are there in making up descriptive titles? What difficulties are involved...
2	How can actually pertinent data, as opposed to references or entire articles themselves, be retrieve...
3	What is information science? Give definitions where possible. ...
4	Image recognition and any other methods of automatically transforming printed text into computer-rea...
5	What special training will ordinary researchers and businessmen need for proper information manageme...

Table 2 shows a sample of the CISI.QRY file, which contains user queries used for evaluating information retrieval systems. Each entry includes a Query ID and the corresponding question or search request. These queries reflect typical information needs in the field of library and information science, making the dataset useful for testing and benchmarking retrieval performance.

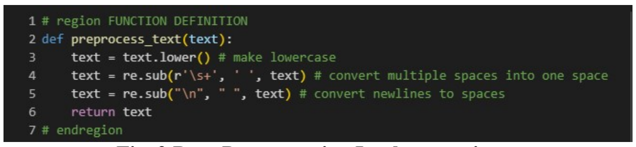
Table 3. Sample content of CISI.REL file

Relevance Judgement Id	Query Id	Relevant Docs Id
1	1	[1281, 650, 1162, 524, 269, 1164, 783, 894, 150, 28]
1	2	[669, 29, 670, 674, 429, 690, 692, 309, 695, 700]
1	3	[640, 131, 1027, 133, 901, 136, 138, 140, 909, 911]
1	4	[321, 420, 329, 332, 980, 310, 601, 315]
1	5	[642, 648, 137, 1035, 525, 400, 528, 32, 692, 56]

Table 3 displays a sample of the CISI.REL file, which provides relevance judgments used in evaluating information retrieval systems. Each row links a query (identified by a Query ID) to a list of relevant document IDs, based on expert judgment. This mapping is essential for assessing how well a retrieval system can return documents that match user needs.

B. Data Preprocessing

The text data was preprocessed by converting all characters to lowercase, removing punctuation, collapsing newlines and multiple spaces. Figure 2 shows the implementation of the data preprocessing. Lowercase conversion is done using lower() function from python. Next, multiple spaces are then converted to single space using regex re.sub(r'\s+', ' ', text). Finally, the newline character is then converted into spaces. Converting multiple spaces and newline characters into space is to reduce the dataset complexity. Further data preprocessing is not performed to pertain to the context of the sentences because BERT embedding will be used in this research.



```python
# region FUNCTION DEFINITION
def preprocess_text(text):
    text = text.lower() # make lowercase
    text = re.sub(r'\s+', ' ', text) # convert multiple spaces into one space
    text = re.sub("\n", " ", text) # convert newlines to spaces
    return text
# endregion
```

Fig. 2 Data Preprocessing Implementation.

C. Relevance Pair Generation

Query-document pairs were created with binary relevance labels (relevant or not relevant) to serve as training samples for classification. Each pair was treated as an input unit for embedding and modeling stages.

Table 4 shows the sample of question-answer pairs. Relevance judgement is shown in column Relevant represented by 1 (relevant) and 0 (not relevant). In this study, there are a total 6228 pairs with 3114 pairs relevant and 3114 pairs not relevant.

Table 4. Sample content of CISI.REL file

No	Question - Answer Pair	Relevant
1	(What problems and concerns are there?, Relative Effectiveness of Document…)	1
2	(What problems and concerns are there?, Information Value of VINITI-published…)	0
3	(What problems and concerns are there?, Relative Effectiveness of Titles, Abstracts, and Subject …)	1
4	(What problems and concerns are there?, Principles of Selective Information Servicing…)	0
5	(What problems and concerns are there?, On basic features of information retrieval…)	1

D. Feature Extraction.

This step transforms each query-document pair into numerical feature vectors. The text is independently embedded using two methods: TF-IDF for statistical term weighting and BERT for semantic contextual representation. After vectorization, the extracted features are combined to be used for classification models. The following is the details of the feature extraction process.

- TF-IDF Vectorization: Each query and document was independently transformed into a 5,000-dimensional sparse vector using TfidfVectorizer. The max_feature parameter is set to 5000 to decrease the complexity of the dataset and increase model performance. This will limit the vocabulary size to 5,000 and enough to capture most frequent terms across the dataset. Other tests using higher max_feature shows that the performance is not better than max_feature 5000. The vectors of queries and documents were concatenated and formed a 10,000-dimensional feature vector. Figure 3 shows the implementation of TF-IDF vectorization.
- Dimensionality Reduction: To align with BERT's vector size and minimize computational load, Truncated SVD was applied, reducing the TF-IDF vector to 768 dimensions. Figure 4 shows the implementation of dimensionality reduction of the TF-IDF vector.
- BERT Embedding: Contextual embeddings for queries and documents were generated using the all-MiniLM-L6-v2 model from SentenceTransformers library. Each text was encoded into a 384-dimensional vector. Queries and

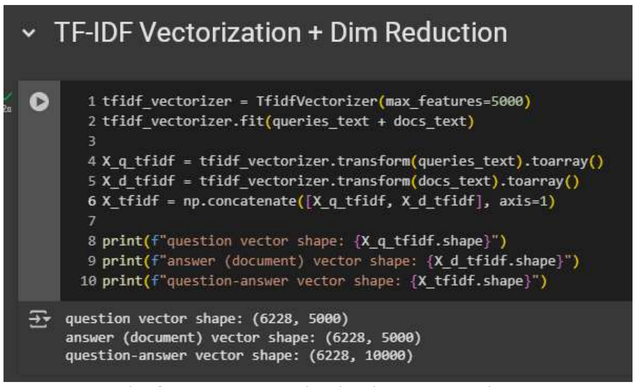
documents vectors are concatenated and create a 768-dimensional vector. Figure 5 shows the implementation of BERT embedding.

- Feature Combination: The reduced TF-IDF vectors and BERT embeddings were combined by using weighting. The following is a formula to combine TF-IDF and BERT vectors.

$$combined\_vector = bw \; x \; bv + tw \; x \; tv \qquad (1)$$

where:
- *bw*: weight of BERT
- *bv*: BERT vector
- *tw*: weight of TF-IDF
- *tv*: TF-IDF vector.

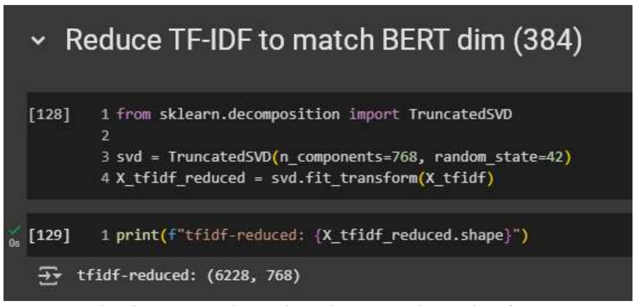
In this research BERT weight is set to 0.9 and TF-IDF is set to 0.1. Based on experiment, this combination gives the best result. Figure 6 shows the implementation of feature combination.



```
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
tfidf_vectorizer.fit(queries_text + docs_text)

X_q_tfidf = tfidf_vectorizer.transform(queries_text).toarray()
X_d_tfidf = tfidf_vectorizer.transform(docs_text).toarray()
X_tfidf = np.concatenate([X_q_tfidf, X_d_tfidf], axis=1)

print(f"question vector shape: {X_q_tfidf.shape}")
print(f"answer (document) vector shape: {X_d_tfidf.shape}")
print(f"question-answer vector shape: {X_tfidf.shape}")
```

```
question vector shape: (6228, 5000)
answer (document) vector shape: (6228, 5000)
question-answer vector shape: (6228, 10000)
```

Fig. 3 TF-IDF vectorization implementation.

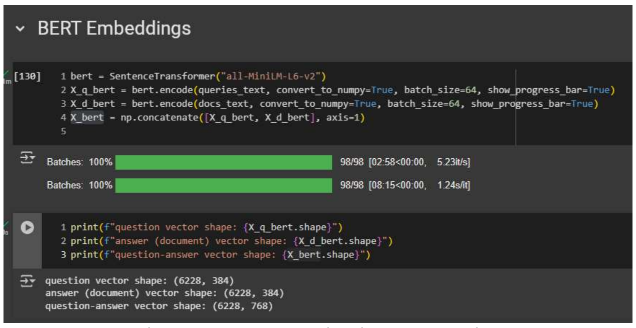


```
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=768, random_state=42)
X_tfidf_reduced = svd.fit_transform(X_tfidf)
```

```
print(f"tfidf-reduced: {X_tfidf_reduced.shape}")
```

```
tfidf-reduced: (6228, 768)
```

Fig. 4 TF-IDF dimensionality reduction using SVD



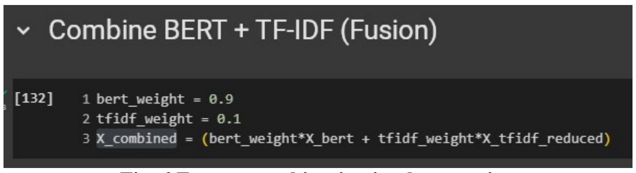
```
bert = SentenceTransformer("all-MiniLM-L6-v2")
X_q_bert = bert.encode(queries_text, convert_to_numpy=True, batch_size=64, show_progress_bar=True)
X_d_bert = bert.encode(docs_text, convert_to_numpy=True, batch_size=64, show_progress_bar=True)
X_bert = np.concatenate([X_q_bert, X_d_bert], axis=1)
```

```
Batches: 100% 98/98 [02:58<00:00,  5.23it/s]
Batches: 100% 98/98 [08:15<00:00,  1.24s/it]
```

```
print(f"question vector shape: {X_q_bert.shape}")
print(f"answer (document) vector shape: {X_d_bert.shape}")
print(f"question-answer vector shape: {X_bert.shape}")
```

```
question vector shape: (6228, 384)
answer (document) vector shape: (6228, 384)
question-answer vector shape: (6228, 768)
```

Fig. 5 BERT embedding implementation

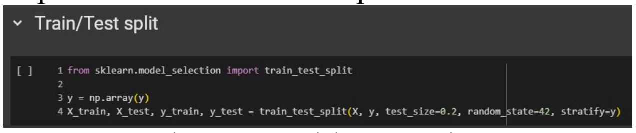


```
bert_weight = 0.9
tfidf_weight = 0.1
X_combined = (bert_weight*X_bert + tfidf_weight*X_tfidf_reduced)
```

Fig. 6 Feature combination implementation

E. Dataset Split

This step split the dataset into test and training data. In this research the dataset is split into 80% training data and 20% testing data. The train_test_split function from sklearn is used to split the data. Stratify parameter is used to maintain the same proportion of labels in training and testing data. Figure 7 shows the implementation of dataset split.



```
from sklearn.model_selection import train_test_split

y = np.array(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

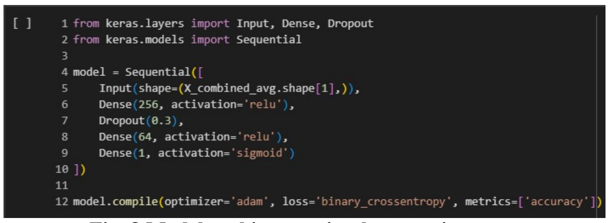
Fig. 7 Dataset split implementation

F. Model Architecture Development

The model is developed using Fully Connected Neural Network (FCNN) that contains several layers. Figure 8 shows the implementation of the architecture. The architecture consists of the following layers:

1. **Input layer**: Accepts input of shape equal to the dimension of the feature vector. In the research, it accepts 1,536 features (768 from TF-IDF + 768 from BERT).
2. **First hidden layer**: A dense layer with 256 neurons.It uses the ReLU (Rectified Linear Unit) activation function to introduce non-linearity.
3. **Dropout layer**: A Dropout layer with a dropout rate of 30%. It helps to prevent overfitting by randomly disabling neurons during training.
4. **Second hidden layer**: A dense layer with 64 neurons, also using ReLU activation.

5. **Output layer**: A dense layer with 1 neuron and sigmoid activation. Outputs a probability between 0 and 1, suitable for binary classification (relevant vs. not relevant).

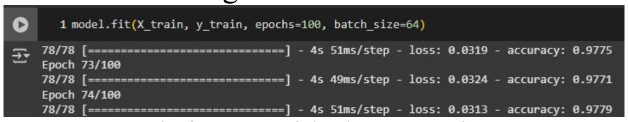


```
[ ]
from keras.layers import Input, Dense, Dropout
from keras.models import Sequential

model = Sequential([
    Input(shape=(X_combined_avg.shape[1],)),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Fig. 8 Model architecture implementation

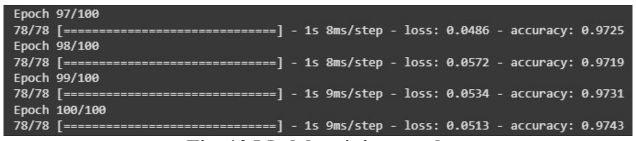
G. Model Training

The fully connected neural network was trained using the processed and labeled query-document pairs. The model was trained for 100 epochs with a batch size of 64. Throughout the training, the model achieved training accuracy of 97.43% in the final epoch. Figure 9 shows the model training implementation and Figure 10 shows the training result.



```
model.fit(X_train, y_train, epochs=100, batch_size=64)

78/78 [==============================] - 4s 51ms/step - loss: 0.0319 - accuracy: 0.9775
Epoch 73/100
78/78 [==============================] - 4s 49ms/step - loss: 0.0324 - accuracy: 0.9771
Epoch 74/100
78/78 [==============================] - 4s 51ms/step - loss: 0.0313 - accuracy: 0.9779
```

Fig. 9 Model training implementation

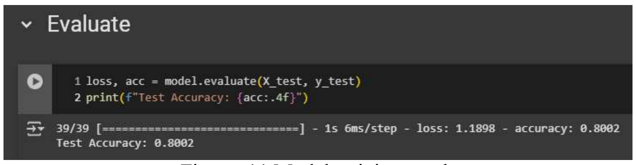


```
Epoch 97/100
78/78 [==============================] - 1s 8ms/step - loss: 0.0486 - accuracy: 0.9725
Epoch 98/100
78/78 [==============================] - 1s 8ms/step - loss: 0.0572 - accuracy: 0.9719
Epoch 99/100
78/78 [==============================] - 1s 9ms/step - loss: 0.0534 - accuracy: 0.9731
Epoch 100/100
78/78 [==============================] - 1s 9ms/step - loss: 0.0513 - accuracy: 0.9743
```

Fig. 10 Model training result

H. Model Evaluation

After training, the model's performance was evaluated using the test set, which consisted of 20% of the total data. The evaluation measured both the loss and the classification accuracy. The final evaluation achieve a test accuracy of 0.8002, indicating that the model correctly predicted the relevance of query-document pairs in approximately 80% of cases. This result demonstrates that the model generalized well to unseen data and did not suffer from overfitting. Figure 11 shows the implementation of model evaluation and its result.



Evaluate

```
1 loss, acc = model.evaluate(X_test, y_test)
2 print(f"Test Accuracy: {acc:.4f}")

39/39 [==============================] - 1s 6ms/step - loss: 1.1898 - accuracy: 0.8002
Test Accuracy: 0.8002
```

Figure. 11 Model training result

III. Results And Discussion

This study evaluated the performance of three approaches in information retrieval tasks: TF-IDF, BERT, and a combination of TF-IDF and BERT. The evaluation was conducted based on accuracy performance metrics for both training and testing datasets. The experiment was conducted using different types of vectorization which are TF-IDF only, BERT only, and combination of TF-IDF and BERT.

Table 1. Accuracy Result

Method	Training	Testing
TF-IDF	0.9777	0.7376
BERT	0.9671	0.7777
TF-IDF + BERT	0.9743	0.8002

Table 1 shows the result of TF-IDF, BERT, and TF-IDF+BERT result. The TF-IDF method achieved a high training accuracy of 0.9777 but showed a lower testing accuracy of 0.7376, indicating potential overfitting and limitations in generalizing to unseen data. In contrast, the BERT model demonstrated more balanced performance with a training accuracy of 0.9671 and a higher testing accuracy of 0.7777, suggesting stronger generalization capabilities due to its contextual understanding of language.

The fusion approach, which combined the strengths of both TF-IDF and BERT, delivered the most optimal results with a training accuracy of 0.9743 and the highest testing accuracy of 0.8002. This indicates that integrating lexical-based statistical features from TF-IDF with semantic-rich embeddings from BERT can enhance model robustness and performance on diverse data.

These findings confirm the original objective of the study—to explore whether a hybrid approach could outperform individual models in information retrieval tasks. The results scientifically support the hypothesis that feature-level fusion enables the model to leverage both local and contextual textual representations, leading to improved accuracy.

Compared to methods which focus on either traditional TF-IDF-based models or deep learning models like BERT separately, our findings demonstrate that a synergistic combination can have good results. While BERT alone has been shown to outperform classical

e-ISSN: 2614-8404
p-ISSN: 2776-3234

methods in many NLP tasks, the addition of TF-IDF still provides valuable discriminative features, especially in domains with sparse or keyword-driven data.

In summary, the combination method between TF-IDF and BERT provides practical implications for improving retrieval performance in real-world applications.

IV. Conclusion

This research aimed to enhance information retrieval performance by combining traditional TF-IDF features with deep contextual representations from BERT. The proposed combination approach successfully outperformed individual models, demonstrating that integrating lexical and semantic information leads to improved generalization and classification accuracy. This work advances the current state of knowledge by offering a hybrid architecture that leverages the complementary strengths of statistical and deep learning methods, providing a more robust solution for text classification tasks. The findings highlight the potential of feature fusion in real-world information retrieval systems, particularly in applications where both keyword specificity and contextual understnding are crucial. Future work may explore integrating additional feature extraction methods or using attention mechanisms to dynamically weight the contribution of each representation, as well as validating the approach on larger and more diverse datasets to further assess scalability and adaptability.

References

[1] Khattab, O., & Zaharia, M. (2020). ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT (*SIGIR '20.*). https://doi.org/10.1145/3397271.3401075

[2] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: a heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv (Cornell University). Retrieved from https://arxiv.org/pdf/2104.08663

[3] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1810.04805

[4] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). ROBERTA: A robustly optimized BERT pretraining approach. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1907.11692

[5] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., . . . Yih, W. (2020). Dense passage retrieval for Open-Domain question answering. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2004.04906

[6] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). https://doi.org/10.18653/v1/d19-1410

[7] MacAvaney, S., Yates, A., Cohan, A., & Goharian, N. (2019). CEDR: Contextualized Embeddings for Document Ranking. SIGIR'19: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieva. https://doi.org/10.1145/3331184.3331317

[8] Nogueira, R., & Cho, K. (2019). Passage Re-ranking with BERT. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1901.04085

[9] Yang, W., Zhang, H., & Lin, J. (2019). Simple applications of BERT for ad hoc document retrieval. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1903.10972

[10] Lin, J., Nogueira, R., & Yates, A. (2020). Pretrained transformers for text

ranking: BERT and beyond. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2010.06467

[11] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021b). BEIR: a heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv (Cornell University). Retrieved from https://arxiv.org/pdf/2104.08663

[12] Gao, L., Callan, J. (2021). Unsupervised Corpus Aware Language Model Pretraining for Dense Retrieva. (*ACL Findings*). https://doi.org/10.18653/v1/2021.findings-acl.413

[13] Reimers, N., & Gurevych, I. (2020). Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation (EMNLP 2020). https://doi.org/10.18653/v1/2020.emnlp-main.733

[14] Gao, L., & Callan, J. (2021). Unsupervised Corpus Aware Language Model Pretraining for Dense Retrieval (*Proceedings of ACL 2021*). https://doi.org/10.18653/v1/2021.acl-long.455

[15] Zhan, J., Mao, Y., Liu, J., Callan, J., & Gao, W. (2021). Optimizing Dense Retrieval Model Training with Hard Negatives. *Proceedings of SIGIR '21*. https://doi.org/10.1145/3404835.3462893

[16] Qu, Y., Liu, L., Liu, J., Ren, P., Chen, Z., Ma, J., & de Rijke, M. (2021). RocketQA: An Optimized Training Approach to Dense Passage Retrieval (*Proceedings of NAACL 2021*). https://doi.org/10.18653/v1/2021.naacl-main.466

[17] Xiong, L., et al. (2021). Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. (*ICLR 2021*). https://openreview.net/forum?id=ze8wPjDclz

[18] Ma, X., Yu, J., & Sun, M. (2022). Multiview Training for IR: Improving Dense Retrieval with Multi-granularity Views. *Proceedings of the Web Conference 2022 (WWW '22)*. https://doi.org/10.1145/3485447.3511970

[19] Hofstätter, S., Reimer, N., & Hanbury, A. (2021). Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling (*ECIR 2021*). https://doi.org/10.1007/978-3-030-72113-8_9

[20] Glasgow IDOM - CISI Collection. (n.d.). [Dataset]. Retrieved from https://ir.dcs.gla.ac.uk/resources/test_collections/cisi/