Semantic-BERT and semantic-FastText models for education question classification

Teotino Gomes Soares [a,b,1], Azhari Azhari [b,2,*], Nur Rohkman [b,3]

[a] Department of Computer Science, Faculty of Engineering & Science, Dili Institute of Technology, Dili, Timor Leste
[b] Department of Computer Science and Electronics, Faculty of Natural Science, Universitas Gadjah Mada. Yogyakarta, Indonesia
[1] tyosoares@gmail.com; [2] arisn@ugm.ac.id; [3] nurrokhman@ugm.ac.id
* corresponding author

ARTICLE INFO

Article history
Received January 23, 2025
Revised May 7, 2025
Accepted May 14, 2025
Available online May 31, 2025

Keywords
Question classification
Semantic parsing
S-Bert
S-FastText
K-fold cross-validation

ABSTRACT

Question classification (QC) is critical in an educational question-answering (QA) system. However, most existing models suffer from limited semantic accuracy, particularly when dealing with complex or ambiguous education queries. The problem lies in their reliance on surface-level features, such as keyword matching, which hampers their ability to capture deeper syntactic and semantic relationships in the question. This results in misclassification and generic responses that fail to address the specific intent of prospective students. This study addresses this gap by integrating semantic dependency parsing into Semantic-BERT (S-BERT) and Semantic-FastText (S-FastText) to enhance question classification performance. Semantic dependency parsing is applied to structure the semantics of interrogative sentences before classification processing by BERT and FastText. A dataset of 2,173 educational questions covering five question classes (5W1H) is used for training and validation. The model evaluation uses a confusion matrix and K-Fold cross-validation, ensuring robust performance assessment. Experimental results show that both models achieve 100% accuracy, precision, and recall in classifying question sentences, demonstrating their effectiveness in educational question classification. These findings contribute to the development of intelligent educational assistants, paving the way for more efficient and accurate automated question-answering systems in academic environments.

1. Introduction

Most contemporary question-answering (QA) systems used in university admissions lack the ability to accurately classify and interpret diverse education questions, particularly those framed using the 5W1H structure (what, how, where, when, who, why). These systems typically depend on static information retrieval techniques, which are insufficient for understanding the semantic intent and structure of domain-specific queries. As a result, they often generate generic, irrelevant responses that do not address the specific concerns of prospective students. This gap in semantic understanding highlights the urgent need for a more intelligent and adaptive question classification approach, one that can accurately analyze and categorize complex questions to enable more personalized and context-aware responses.

Although advancements in natural language processing (NLP) and deep learning have enhanced question classification in various domains, models like BERT [1], [2] SS-BERT [3], LSTM [4], LLM [5]–[9] and Transformer [10]–[13] have yet to be optimized for the complexities of university

admissions. Existing methods often overlook the unique linguistic structures of admission-related inquiries, which limits their effectiveness.

Several recent studies highlight these limitations, such as those of Gweon et al. [1], who applied BERT for open-ended question classification, achieving an 86% accuracy rate on two different datasets. Fu et al. [3] introduced SS-BERT for adversarial argument selection in open-domain QA, obtaining 70% accuracy. Al Faraby et al. [14] explored BERT, XLNet, and RoBERTa for categorizing questions into ten cognitive science categories, reporting 84% accuracy for BERT and 95% for RoBERTa. Xiao et al. [15] applied FastText to classify Mandarin legal domain questions, achieving 95.75% accuracy.

While these approaches demonstrate improved classification performance, they do not explicitly model the semantic dependencies within question structures, which is crucial for accurately distinguishing intent variations in similar-looking questions. Traditional machine learning techniques [16], [17], and deep learning-based classifiers [10], [11], [18]–[20] also suffer from limited semantic understanding, making them insufficient for handling complex question classification tasks in the university admissions system

To address these limitations, this research proposes the development of Semantic-BERT (S-BERT) and Semantic-FastText (S-FastText) models, which integrate semantic dependency parsing to enhance question classification accuracy. By modeling word relationships in 5W1H questions, these models improve intent recognition and classification performance.

Thus, the main contributions of the proposed approach are:

- The development of S-BERT and S-FastText models was designed explicitly to classify prospective students' questions in university admissions.
- Integration of semantic dependency parsing to improve classification accuracy by enhancing word relationship understanding.
- Experimental evaluation using training and validation datasets, demonstrating the effectiveness of the proposed models compared to existing methods.
- Structured performance analysis utilizing confusion matrices and K-Fold cross-validation to ensure the models' robustness.

This study comprises an introduction section, which presents ideas about the problem that are related to and build upon previous studies. The method section presents the proposed methodology, including dataset preparation, model development, and integration of semantic dependencies. The results and discussion section presents the experimental findings, comparing S-BERT and S-FastText with existing approaches. The last section, Conclusion, summarizes the key findings and suggests directions for future research.

2. Method

Our proposed question classification model integrates semantic dependency parsing with BERT and FastText. The model development begins with a preprocessing phase that includes tokenization, stop word removal, lowercasing, lemmatization, and manual labeling of each question sentence. Labeling is conducted by reviewing each question and assigning a label based on interrogative words such as what, who, where, when, and how. After labeling, stratified Sampling is applied to ensure balanced label distribution in the split of the training and validation datasets, which consist of 2,175 samples. Both datasets undergo validation using K-Fold cross-validation. Model training performance is evaluated using a confusion matrix to assess accuracy, precision, recall, and F1 score. The validation model utilized K-Fold cross-validation with the validation dataset. The S-BERT and S-FastText models used for educational question classification are illustrated in Fig. 1.

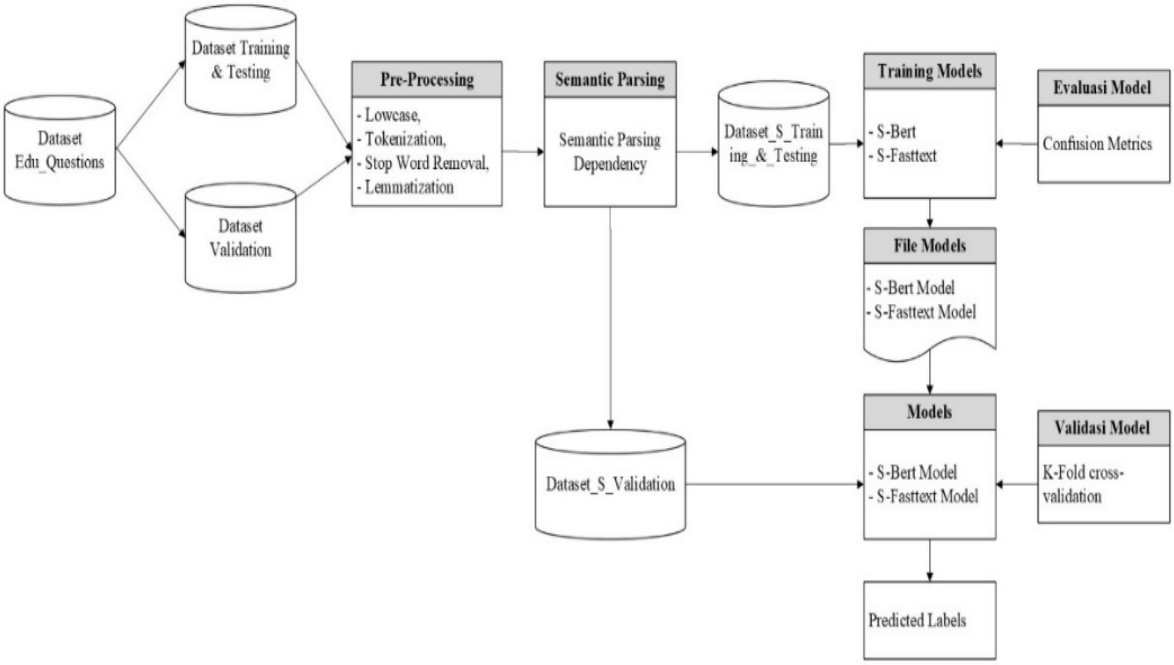


Fig. 1. Edu-question classification process model

2.1. Data Collection

The study utilized data from the frequently asked questions (FAQs) of prospective Dili Institute of Technology (DIT) students. This data is processed through several stages, including preprocessing, labeling, distribution analysis, and validation, to create a dataset of questions from new students. The Dataset was then divided into an 80% training split testing dataset and a 20% validation dataset, with the data distribution outlined in Table 1.

Table 1. Distribution dataset training and the dataset validation

Question Labels	Dataset Training	Dataset Validation
when	311	78
where	290	73
how	294	74
what	272	68
why	296	74
who	276	69
Total	1739	436

The dataset distribution in Table 1 suggests that some question categories have slightly fewer samples than others. To address this, we applied stratified Sampling during the data split for training and validation. This approach preserved the proportional representation of each category, ensuring balanced exposure during model training. Additionally, K-fold cross-validation was employed to further mitigate the risk of class imbalance by training and validating the model on multiple subsets of the data.

2.2. Preprocessing

In the preprocessing, we conducted tokenization, removal of stop words, lowercasing, lemmatization, part-of-speech (POS) tagging, and labeling will be performed on the question data.

- Tokenization segments a question sentence into linguistic units, allowing for a structured representation of words that form essential grammatical elements [21], [22] In this study, we use the spaCy library [23] for tokenization, which effectively handles word boundaries, including sub-words, ensuring better performance in downstream parsing and classification

- Stop Word Removal is used for uninformative words (e.g., *the, is, of*) that contribute little to meaning [21], [22]. By reducing dimensionality, stop-word removal enhances processing efficiency and helps improve classification accuracy. Studies indicate that stop-word removal can reduce word index storage requirements by 30%-50% [22].
- Stemming converts words into their base form using vocabulary-based linguistic analysis (e.g., *running* → *run*, *better* → *good*). This normalization ensures consistency in word representation, allowing models to generalize better across different word variations [23], [22].
- Part of Speech Tagging (POS) is used to assign grammatical categories to words, providing syntactic information essential for dependency parsing and question classification [21], [24]. This study uses Conditional Random Fields (CRF) Tagger from the NLTK library for POS tagging. Standard parts of speech include: (1) noun, (2) verb, (3) pronoun, (4) preposition, (5) adverb, (6) conjunction, (7) adjective, and (8) interjection [24], [25].
- Stratified Sampling is used to ensure a balanced label distribution in the Dataset [26], and we apply stratified Sampling when splitting the data into training and validation sets. This method prevents class imbalance, reduces model bias, and improves generalization performance [17], [26].

2.3. Semantic Parsing

In this groundbreaking study, we harness the power of semantic processing to transform natural language sentences into formal logical forms, revolutionizing how computers process information [27]. Syntactic parsing is at the core of semantic processing, which dissects the grammatical structure of sentences using rule-based methods, employing constituency trees and dependency parsing to intricately depict the hierarchical relationships between sentence components [28]. Dependency parsing, focusing on inter-word relationships, designates one word as the "core" while others are referred to as "dependent words" [28], [29].

The semantic dependency graph is represented as a directed graph, with Equation (1) [30] such as

$$G = (V, E, R) \tag{1}$$

Where V is the set of nodes (words in the sentence), E is the set of directed edges representing dependencies, and R is the set of semantic roles (e.g., Subject, Object, Predicate).

Our research applied semantic dependency parsing to all question sentences in the training and testing datasets after preprocessing. Using the en_core_web_sm model for English, this semantic process identifies the root word, prioritizing verbs, and if no verb is present, a noun is used as the root word. A custom tag was also added to classify question words as "QW." The results of this parsing were saved in dataset-s-training_&_testing and dataset-s-validation for use in model training and testing, with the outcomes illustrated in Fig. 2.

(a) (b)

Fig. 2. Semantic dependency parsing sentences (a) question data Training (b) question data testing

Fig. 2 shows that semantic dependency parsing captures the hierarchical relationships between words, enabling a structured representation of question semantics. Unlike conventional syntactic parsing, semantic dependency parsing focuses on functional dependencies that determine question intent, which

is particularly relevant for classifying 5W1H questions in educational contexts. This structure allows the model to capture the semantic roles and relationships accurately, enhancing intent recognition and classification. To generate the output depicted in Fig. 2, it is crucial to develop an algorithm that utilizes Python libraries such as spaCy and classifies interrogative words ("where," "who," "why," "what," "when," "how") as question words (QW). An essential phase in Algorithm 1 is the processing of sentences using spaCy (lines 5-23). Each Token undergoes analysis in this phase, interrogative words are marked as QW, and the dependencies between tokens are constructed and stored for future model training. For additional information, please refer to Algorithm 1.

Algorithm 1: Build a semantic parsing dependency
1. **Load** the spaCy model
2. **If** not hasattr(Token, 'custom_pos') **then**:
3. Register a custom extension 'custom_pos' for Token
4. **Define** a set of question words: W= {"where", "who", "why", "what", "when", "how"}
5. **Function** process_sentence(sentence):
6. Process the sentence using spaCy to create a doc object
7. Initialize custom_data dictionary:
8. custom_data = {'words': [], 'arcs': []}
9. **For** each token t in the doc, **do**: i
10. **If** t.text.lower() in W **then**:
11. Assign custom POS tag 'QW' to t._.custom_pos
12. **Else**:
13. Assign default POS tag to t._.custom_pos
14. **End if**
15. **Add** dictionary {'text': t.text, 'custom_pos': t._.custom_pos} to custom_data['words']
16. **For** each token t in the doc, **do**: i
17. **If** t.head is not t, **then**:
18. Create an arc dictionary with 'start', 'end', 'label', and 'dir' based on token positions and dependencies
19. Append arc to custom_data['arcs']
20. **End if**
21. **End for**
22. **Optional**: Render the dependency tree using display with custom POS tags
23. **Return** custom_data['words']
24. **Load** the dataset from the CSV file
25. **Ensure** the column text_lemmatized exists in the dataset
26. **Create** new columns parsing_result and parsing_question in the dataset
27. **For** each row in the dataset, **do**:

2.4. Fast-text Model

Fast-text is an extension of the popular word embedding model word2vec, developed by the Facebook research team [31] and inspired by earlier research findings [32]. Known for its impressive capability to train on 1 billion words in just 10 minutes, Fast-text achieves high accuracy compared to other models. A notable feature of Fast-text is its use of n-gram sub-words for word embedding, which allows it to effectively handle out-of-vocabulary (OOV) words and generate corresponding vectors [33], [34]. Its architecture mirrors the continuous bag-of-words (CBOW) structure of word2vec, consisting of three layers: the input, hidden, and output layers [35].

In this research, the outcomes of semantic parsing dependencies are utilized as input through the input layer of the Fast-text model. These inputs are then processed in the hidden layer, where the model analyzes the data to classify question labels. The results of this processing are subsequently presented through the output layer, demonstrating the model's capability in question label classification. The architecture of the S-Fast-text model is illustrated in Fig. 3.

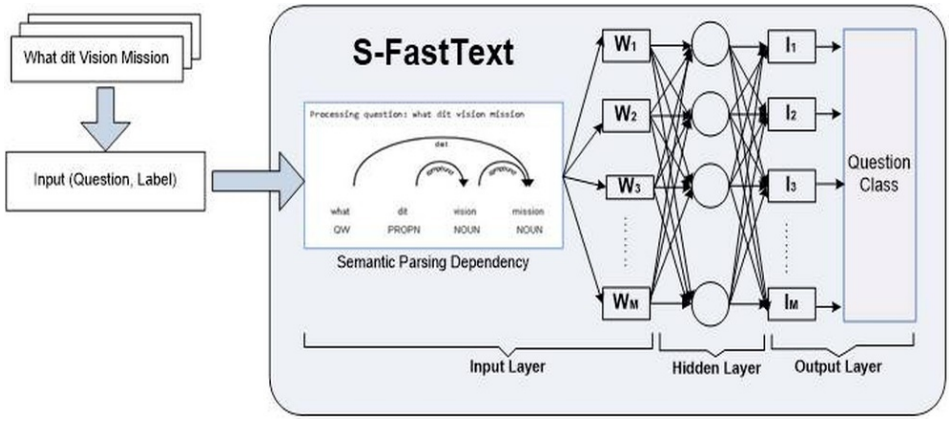


Fig. 3. S-FastText model architecture for Edu-question Classification

The input of this model is a question in text form, such as "What is DIT's vision mission?", and labels related to the classification of the question. Before the question is processed, semantic parsing is performed to identify the syntactic and semantic relationships between the words in the question. This parsing produces the dependency structure of the sentence. For example, the word "what" is marked as "QW" (question word), "dit" as PROPN (Proper Noun), "vision" and "mission" as NOUN (Noun).

After parsing, the results are fed into the input layer to be converted into a vector representation of the question sub-words using the Fast-text embedding method. The vector representation values are passed to the hidden layer, containing several neurons to process information from the word vector and map it to a more abstract representation. The interpretation results from the hidden layer are strategically directed toward the Output Layer, where they culminate to deliver the final prediction in the form of the question class.

Calculate probability values using layered soft-max based on the Huffman tree, where each leaf node represents a text category. Each leaf node selects the highest probability as the target category [36] using Equation (2).

$$p(w_i) = \prod_{j=1}^{L(w)-1} \sigma\left(sign(w_i, j).\theta_{n(w_i,j)}^{T} h\right) \quad (2)$$

Where, $\theta_{n(w_i,j)}^{T}$ represents the vector of non-leaf nodes $n(w_i, j)$ The output vector h Represents the output value of the hidden layer, which is calculated from the input word vector. $sign(w_i, j)$ Represents a particular function whose Value is $\{-1,1\}$.

To implement the architecture of the S-FastText model shown in Fig. 3. It is necessary to develop a suitable algorithm for performing question classification, as outlined in Algorithm 2.

Algorithm 2: Build S-Fasttext model
1. **Input**: File CSV
2. **Read** the CSV file
3. **Extract** the 'parsing_question' column to list texts.
4. **Extract** the 'label_tags' column to list labels.
5. **Open** file "formatted_datatrain.txt" for writing
6. **For** each text and label in texts and labels, **do**:
7. Format the line as "label{label} {text}"
8. Write the formatted line to "formatted_datatrain.txt"
9. **End for**
10. **Close** file "formatted_datatrain.txt"
11. **Initialize** model S-FastText with parameters:
12. Learning rate = 0.3
13. Epoch = 10
14. Word N-grams = 2
15. Train the model using data from" formatted_datatrain.txt"

Save the trained model as "S-Fasttext-Model.bin."

In Algorithm 2, hyperparameters such as the number of epochs, learning rate, and N-grams should be set to their maximum values to ensure the model achieves optimal performance. The hyperparameter settings for pre-training the S-FastText model can be found in Table 2.

Table 2. Hyperparameter setting for pre-training S-FastText model

Model	Epoch	Learning Rate	n-gram
S-FastText	10	0.3	2

Hyperparameter setting for pre-training S-FastText model on Table 2. using epochs 10 to balance training time and model generalization. Learning Rate 0.3 for faster convergence. For N-grams, the Bigram setting enhances sub-word-level feature extraction.

2.5. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model designed to consider the context of words from the left and right sides simultaneously, with a simple conceptual [36]–[38]. This study uses a large BERT model consisting of 24 transformer encoder blocks and 16 self-attention heads, trained with a hidden size of 1024 and a maximum token sequence length of 512. This model has around 340 million parameters [37], [39]. This large BERT model will be modified by adding dependency parsing semantics to create the S-BERT model as shown in Fig. 4. The Architecture of S-BERT in this research consists of a dependency parsing semantics process, an input embedding layer, a transformer encoder layer, and an output layer.

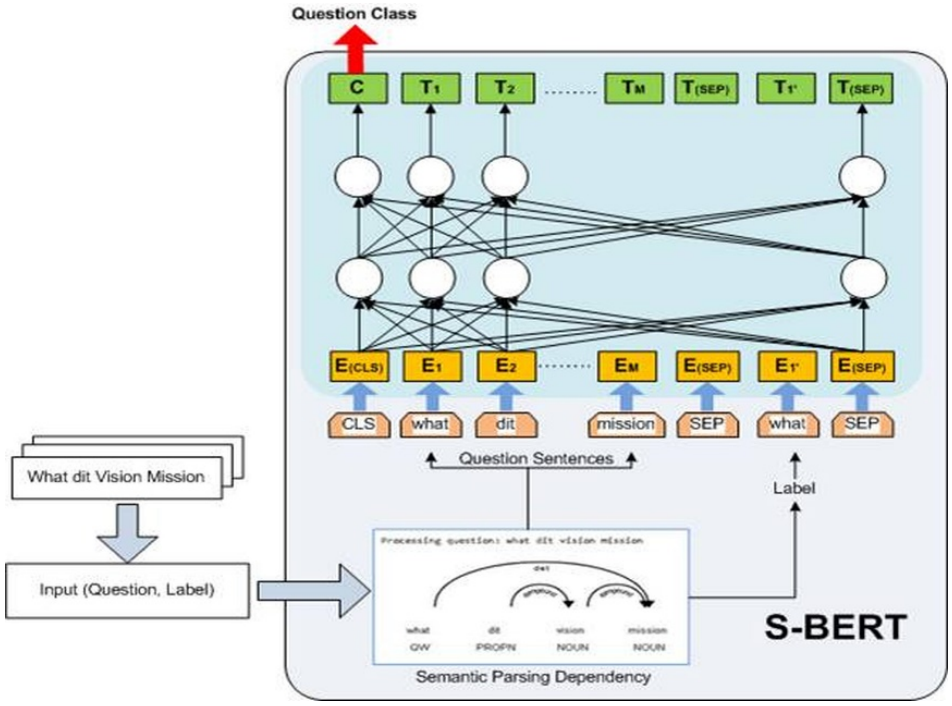


Fig. 4. S-BERT architecture model for Edu-question classification

In the process of dependency parsing semantics, a question sentence is analyzed to identify its semantic structure and dependency relationships. For example, the sentence "What is the vision and mission of the Department?" is analyzed to determine the question word (QW), ROOT, NOUN, and PROPN (proper noun). This is done after going through the preprocessing stage of the sentence. The dependency parsing semantics results are then processed through an input embedding layer involving token embedding, segmentation embedding, and position embedding. The set of tokens processed through these three embedding layers, with the exact dimensions, is then added together and passed to the encoder layer [38].

Each Token (x_2) embedding in BERT is represented using Equation (3), such as:

$$E_i = T_i + P_i + S_i \tag{3}$$

where T_i is token embedding, P_i is position embedding and S_i Is segment embedding:

The input embedding result will be processed by the Transformer Encoder Layer, as seen in Fig. 5(a). This layer consists of a sub-layer with simple attention, a sub-layer with fully connected feedforward, and a normalization layer as the output of each sub-layer with Layer-Norm ($x + sublayer(x)$) [39]. Sub-layer (x) is the function implemented by the sub-layer itself [37].

The term "attention" can be described as a function that maps a query and a set of key-value pairs to an output, where the query (Q), key (K), Value (V), and output are all vectors. The output is calculated as the weighted sum of the values (V), with the weights given by the query (Q)and the corresponding key. (K) According to a compatibility function, as seen in Fig. 5(b). The attention sub-layer has multi-head self-attention functions, including scaled multi-head self-attention and dot-product attention. Scaled multi-head self-attention uses an attention function with the $d_{model} - dimension$ of $K, V,$ and Q, used for linear projection. Each $Q, V,$ and K, with iteration h, a different linear projection with the dimensions of d_k, d_q, and d_v. Each version of the projected (Q), (K), and (V) then runs the attention function in parallel to produce an output value of the combined dimension. d_v, which is then projected again to produce the final Value. The average Value of one attention head is calculated using Equation (4) [39].

$$MultiHead\ (Q, K, V) = Concat(head_1, \dots, head_n)W^o \tag{4}$$

where $head_i = Attention(QW_i^Q,\ KW_i^K,\ VW_i^V)$ with the matrix projection of each parameter $QW_i^Q \in R^{d_{model}\ x\ d_q}$, $KW_i^K \in R^{d_{model}\ x\ d_k}$, $VW_i^V \in R^{d_{model}\ x\ d_v}$ dan $W^o \in R^{hd_v\ x\ d_{model}}$

The scaled Dot-Product Attention is the primary process of the multi-head self-attention layer in Fig. 5(c). Its input consists of query dimension (d_k) and Value dimension (d_v). To calculate the scaled dot-product attention, we take the dot product of the query with all the Keys, divide by the square root of $\sqrt{d_k}$, and apply the softmax function to obtain weights for each Value. The attention function for a set of queries is grouped into a matrix Q, with the keys as a matrix K, and the values can be computed using Equation (5) [39].

$$Attention\ (Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

In addition to the attention sub-layer, each layer in the encoder and decoder construction contains a Feedforward Network as a connector that will be applied to each position separately. This network consists of a linear transformation and ReLU activation. The Value of the Feedforward Network can be obtained through Equation (6) [39].

$$FFN(x) = \max(0, xW_1 + b_1)\,W_2 + b_2 \tag{6}$$

where $FFN(x)$ is the feedforward network value function, $max(0, x)$ is the maximum function, and $\boldsymbol{x}$ is the input value.

From the S-Bert process, we obtain a question label based on the semantic context of the sentence. This label is classified according to the question's semantic context and then processed by the S-Bert model, which refers to the category or classification of the question asked. To implement the architecture of the S-Bert model shown in Fig. 5.

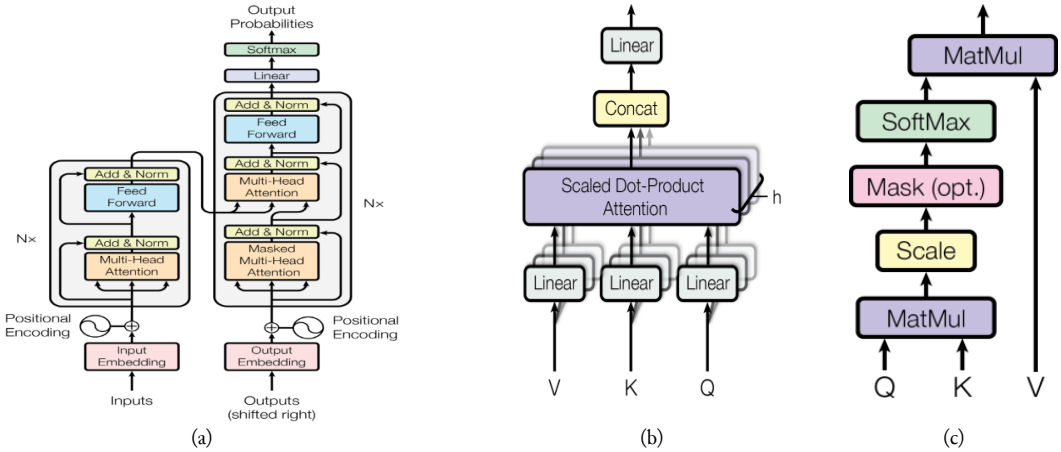


Fig. 5. Transformer Encoder Layer (a), Multi-Head Attention, (b) and Dot-Production Attention (c) [40]

Fig. 5, it is necessary to develop a suitable algorithm for performing question classification, as outlined in Algorithm 3.

Algorithm 3: Fine-tuning S-BERT for Question Classification

1. **Data**: Training set $S = (q, l)$, q the question text of token length w, l the ground truth type sequence for q, of length w.
2. BERTBASE transformers *S-BERT*, with pre-trained model parameters$\theta_T = [\theta_{qT,1}.. \theta_{qT,L}]$, Linear Classifier C with model parameters $\theta_C = [\theta_{\{C,1\}}... \theta_{\{C,L\}}]$, L, L' the respective number of model layers, hyperparameters: learning rate η, *epoch_num*
3. Result: Transformer *S-BERT*, Classifier C with updated parameters θ_T, θ_C respectively
   // Data Preprocessing
4. ***df*** ← Input: File CSV
5. ***label_map*** ← {'what': 0, 'who': 1, 'where': 2, 'when': 3, 'how': 4, 'why': 5}
6. ***df***['question_label'] ← map(*label_map*, *df*['question_label'])
7. ***df*** ← *df*.dropna(subset=['prasing_question', 'question_label'])
8. texts ← *df*['prasing_question'].tolist()
9. label_*tags* ← *df*['question_label'].tolist()
10. **If** *texts* or *label_tags* are empty:
11. Raise ValueError("No data to tokenize. Check the DataFrame processing steps.")
12. **while** *epoch* < *epoch_num* **do**
13. S_{batch} ← sample(S, b) // sample a batch of size b
14. **for** $(q, l) \in S_{batch}$ **do**
15. q' ← tokenize(q)
16. $V_q = F($ *S-BERT*$(q'))$, F the last layer outputs of *S-BERT*
17. $p = C(C_q)$, the predicted type sequence for q
18. $loss = \sum_{i-1}^{w}$ CrossEntropy(l_i, p_i)
19. $\theta_T \leftarrow \theta_T - \eta \nabla loss$
20. $\theta_C \leftarrow \theta_C - \eta \nabla loss$
21. **end for**
    // Print epoch loss
22. **end while**
    // Evaluation
23. Initialize *predictions* and *true_labels* as empty lists.
24. **For** *batch* in *test_loader*:
25. Move *batch* to device.
26. Compute outputs with *model*(*input_ids*, *attention_mask=attention_mask*)
27. Extend *predictions* with argmax(*logits*, dim=1).cpu().tolist()
28. Extend *true_labels* with *labels*.cpu().tolist()
29. **End for**
30. ***accuracy*** ← accuracy_score(*true_labels*, *predictions*)
31. ***precision***, ***recall***, ***f1*** ← precision_recall_fscore_support(*true_labels*, *predictions*, average='weighted')
32. **Print** *accuracy*, *precision*, *recall*, *f1-score*
33. ***model***.save_pretrained('s-bert-question-classifier')
34. ***tokenizer***.save_pretrained('s-bert-question-classifier')
35. ***return*** θ_T, θ_C

Algorithm three above indicates that hyperparameters such as the number of epochs, learning rate, and optimizer must be set to their maximum values to ensure the model achieves optimal performance. The hyperparameter settings for pre-training the S-BERT model can be found in Table 3.

Table 3. Hyperparameter setting for pre-training S-Bert model

Model	Epoch	Learning Rate	Optimizer
S-Bert	10	0,00004	AdamW

Hyperparameter setting for pre-training S-BERT model on Table 3 using epochs 10 to balance training time and model generalization to avoid overfitting and to ensure the model learns efficiently without over-optimizing for the training data. Learning Rate 0.00004 prevents overfitting, is generally more stable, and minimizes the risk of divergence. Optimizer AdamW ensures efficient weight decay.

2.6. Evaluation and Validation Model

2.6.1. Evaluation Model

An evaluation matrix was used to evaluate models:

- A confusion matrix is used to evaluate the classification model [21], [41] to know the precision, recall, F1 measure, and accuracy.
- Precision determines the ratio of relevant retrieved documents to all retrieved documents in the ranking list [41], [42].
- Recall the smallest information retrieval from documents relevant to the successfully retrieved request [41], [42].
- F1 Measure was used to compare the average Value of weighted precision and recall [25] and [41].
- Accuracy in calculating the level of closeness between the predicted Value and the actual Value [43], [25], [41], [42].

2.6.2. Validation Model

K-fold cross-validation is part of the cross-validation method used to validate the performance of a classification machine learning model [44]. Model validation in the k-fold cross-validation dataset is divided into k subsets with the same number: the k-fold training dataset and the k-fold validation dataset. The model is validated with each k-flood-1 up to k-fold [45]. The following are the stages of k-fold cross-validation [44], [46].

- The total data is divided into k parts.
- The 1st fold data becomes the validation data, and the rest becomes the training data. Then, calculate the model performance with the actual numbers or data based on the portion of the data Equation (7)

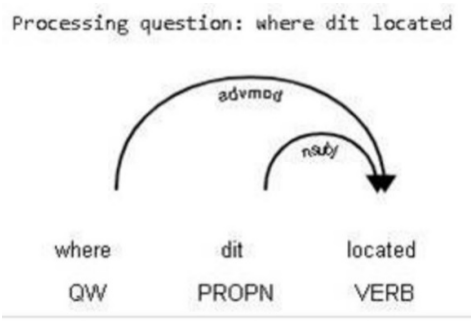
$$Performance = \frac{\sum total\ clasiffication\ data\ k\ flod}{\sum total\ data\ validation}\ x\ 100 \quad (7)$$

- The 2nd fold data becomes the validation data, and the rest becomes the training data. Then, the accuracy is calculated based on the portion of the data.
- With the same process until it reaches the k fold.
- Calculate the average model performance from k to the final model performance

3. Results and Discussion

3.1. Results

The results of the semantic parsing dependency are carried out on all question sentences in Fig. 6(a) and stored in dataset_s_training, as shown in Fig. 6(b). This data will then be utilized in the training and testing process of the S-Bert model.



(a)

stop_text	label_tags	parsing_result	prasing_question
what dit	what	what(QW); dit(NOUN)	what dit
where dit located	where	where(QW); dit(PROPN); located(VERB)	where dit located
what dit vision mission	what	what(QW); dit(PROPN); vision(NOUN); mission(NOUN)	what dit vision mission
what departments dit	what	what(QW); departments(NOUN); dit(VERB)	what departments dit
what faculties dit	what	what(QW); faculties(NOUN); dit(NOUN)	what faculties dit

(b)

Fig. 6. Results of the semantic parsing dependency: (a) Parsing and (b) Parsing results in the Dataset

The same semantic parsing process is applied to the testing dataset to ensure consistency in the testing phase. This step enhances the model's understanding of question intent by leveraging semantic role information in question classification.

3.1.1. Build the S-FastText model

Initialization of the S-FastText model was trained using a dataset split into 80% for training and 20% for validation without using pre-trained embeddings initially. This indicates that the model was trained entirely from scratch. His approach was likely chosen because the Dataset is highly specific to educational questions, allowing the model to better adapt to the language structure and context used in this domain. Additionally, this method helps minimize potential biases that might arise from using pre-trained embeddings trained on general corpora, which may not be relevant to the educational context. Table 4 shows the pre-training performance metrics, and the confusion matrix evaluation is presented in Fig. 7. Model validation is conducted using K-Fold cross-validation, summarized in Table 7.

Table 4. Pre-training S-FastText model performance

Model	Accuracy	Precision	Recall	F1-score
S-FastText	1.00	1.00	1.00	1.00

The confusion matrix results indicate 100% accuracy across all question categories, demonstrating the model's effectiveness in distinguishing between different types of questions. This high performance is attributed to semantic dependency parsing, enhanced understanding of question structure and intent, N-gram Sub-word embeddings, which improved the handling of out-of-vocabulary (OOV) words and morphological variations, and the Use of a Balanced Dataset and Stratified Sampling, Which minimized model bias and enhanced generalization.

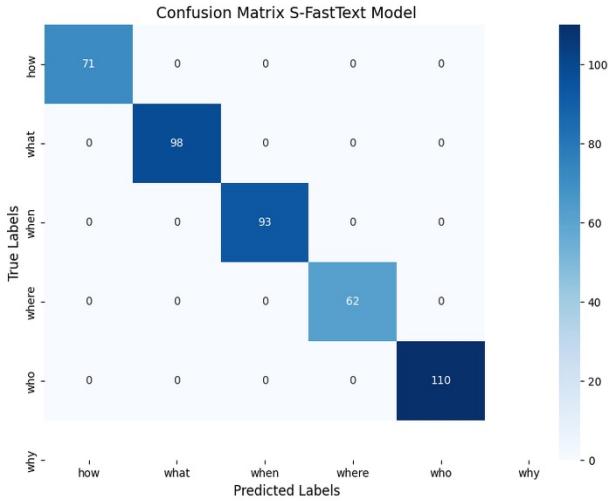


Fig. 7. Confusion Matrix Evaluation Results for the Pre-training S-FastText Model

3.1.2. Build the S-Bert model

The S-Bert model is trained using the same 80%-20% data split, with hyperparameters optimized for optimal performance. Table 5 presents the pre-training metrics, and the confusion matrix evaluation is shown in Fig. 8. K-fold cross-validation is used for model validation, with the results summarized in Table 7.

Table 5. Pre-training S-Bert model performance

Model	Accuracy	Precision	Recall	F1-score
S-Bert	1.00	1.00	1.00	1.00

The S-BERT model achieves 100% accuracy across all question categories, indicating its superior capability in understanding question semantics. This exceptional performance is due to deep contextual embeddings being an accurate semantic representation of words and phrases, Semantic dependency integration enhanced comprehension of hierarchical sentence structures, and transformer architecture adequately contextualized question components.

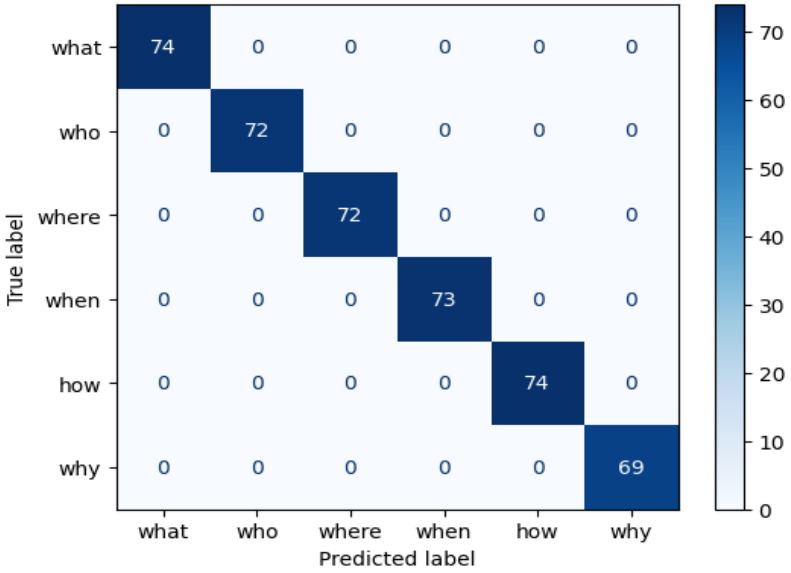


Fig. 8. Confusion Matrix Evaluation Results for the Pre-training S-Bert Model

3.1.3. Validation model

Both models were validated using K-Fold cross-validation with k folds = 5 on the validation dataset. A detailed result validation model for each fold is shown in Table 6.

Table 6. Result validation S-FastText and S-Bert Model for Each Fold

Proposed Models	Folds	Accuracy	Precision	Recall
S-FastText	Fold-1	1.00	1.00	1.00
	Fold-2	1.00	1.00	1.00
	Fold-3	1.00	1.00	1.00
	Fold-4	1.00	1.00	1.00
	Fold-5	1.00	1.00	1.00
S-Bert	Fold-1	1.00	1.00	1.00
	Fold-2	1.00	1.00	1.00
	Fold-3	1.00	1.00	1.00
	Fold-4	1.00	1.00	1.00
	Fold-5	1.00	1.00	1.00

The same hyperparameter settings were maintained as in the training phase. Table 7 presents the validation performance metrics.

Table 7. Performance validation of the proposed models

Proposed Models	Accuracy	Precision	Recall	F1-score
S-Bert	1.00	1.00	1.00	1.00
S-FastText	1.00	1.00	1.00	1.00

The validation results confirm the robustness of both models, achieving 100% accuracy, precision, recall, and F1-score. This consistency is attributed to semantic parsing consistency, which ensures uniform semantic role labeling across training and validation datasets. Also, the Cross-validation strategy effectively models generalization and bias minimization.

3.2. Discussion

This study introduces two innovative model approaches, S-BERT and S-FastText, designed explicitly for educational question classification. Our experiments demonstrate that both models perform exceptionally from pre-training to model evaluation, with 100% accuracy, precision, recall, and F1-score. These results indicate the models' robust capability to accurately classify educational questions, outperforming conventional models in the same domain.

Our findings are significantly superior compared to previous studies. For example, Wei et al. [47] utilized a pre-trained BERT baseline to classify COVID-19-related questions into 15 categories, achieving an accuracy of 58.1%. This lower performance can be attributed to the absence of semantic dependency parsing and the challenges posed by the broader and more complex question categories.

In contrast, our proposed models incorporate semantic dependency parsing to capture hierarchical sentence structures and semantic roles, enhancing intent recognition and question classification accuracy. Additionally, integrating deep contextual embeddings in S-BERT and sub-word n-gram embeddings in S-FastText contributes to their superior performance. The performance comparison between our proposed models and previous research is illustrated in Table 8.

Table 8. The performance of previous research models with our proposed model

Ref.	Year	Methods	Accuracy
Wei et all. [47]	2020	Bert baseline	58.1%
Proposed model	2025	S-Bert	94.57%
Proposed model	2025	S-FastText	97%

Several factors contribute to the outstanding performance of our proposed models: (a) Semantic Dependency Parsing, which enhances the understanding of hierarchical relationships between words, thereby improving intent recognition for 5W1H questions. (b) Contextual embeddings in S-BERT enable the accurate differentiation of semantically similar questions by capturing nuanced word meanings. (c) N-gram Sub-word Embeddings in S-FastText to Improve handling of out-of-vocabulary (OOV) words and morphological variations. (d) Balanced Dataset and Stratified Sampling to ensure consistent label distribution, minimize model bias, and enhance generalization. (e) Hyperparameter settings for S-BERT are learning rate = 2e-5, batch size = 8, and optimizer using AdamW, and S-FastText is learning rate = 0.5, batch size = 8, and wordNgrams=2.

While the models achieved perfect metrics, certain limitations were identified, such as high accuracy, which raises concerns about the risk of overfitting. K-fold cross-validation and early stopping were employed to mitigate this issue, but further testing on more extensive and diverse datasets is necessary. The models were evaluated on a specific dataset of educational questions. Their applicability to other academic contexts or domains remains to be validated.

Future work will involve testing external datasets to evaluate the models' generalization capability beyond the training domain. Explore data augmentation techniques such as paraphrasing and SMOTE

(Synthetic Minority Over-sampling Technique). Ablation studies will also be conducted to assess the contribution of semantic dependency parsing to model performance. Learning curves of training and validation sets will be analyzed to address concerns about overfitting to monitor performance consistency. Regularization techniques, including Dropout and L2 regularization, will be explored to enhance model generalization.

Based on the models' limitations, Future research should focus on expanding the Dataset to more extensive and diverse educational question datasets to enhance robustness and generalizability. Multilingual capability to extend the models to support multilingual educational queries, improving their adaptability in international educational contexts. Hybrid Models to integrate semantic parsing with advanced transformer architectures like T5 or GPT-3 to further improve semantic understanding, and implementing the models in educational platforms, like Chatbots and FAQ Systems, for real-world testing and evaluating their impact on student engagement and admissions efficiency.

4. Conclusion

The experimental results demonstrate that classifying educational questions using the S-FastText and S-BERT models achieves 100% accuracy during pre-training and model validation. Additionally, we applied both models to pre-train a COVID-19 question classifier, where their performance surpassed the baseline BERT model used in this study. This highlights the potential of these models to achieve superior performance compared to traditional approaches. This study can serve as a valuable reference for future researchers selecting appropriate models for question classification in question-and-answer systems. However, additional testing with a diverse and larger dataset is recommended to validate the robustness and applicability of the models further. This will help ensure the accuracy and generalizability of both models in real-world applications. Moreover, it would be beneficial to evaluate the models against other machine learning approaches to compare their performance and identify areas for improvement. By exploring these future directions, researchers can continue to refine these models, ultimately enhancing the performance and scalability of question classification systems in various institutions, including multi-label, hierarchical classification with evaluated confidence classification using log-loss or ROC-AUC, and efficient computation.

Acknowledgment

At the end of this research paper, the author would like to thank the Dili Institute of Technology, the accompanying lecturers, and all the families who supported the completion of this scientific research work.

Declarations

Author contribution. All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.
Funding statement. None of the authors has received funding or grants from any institution or funding body for the research.
Conflict of interest. The authors declare no conflict of interest.
Additional information. No additional information is available for this paper.

References

[1] H. Gweon and M. Schonlau, "Automated Classification For Open-Ended Questions With BERT," J. ofSurvey Stat. Methodol., vol. 12, pp. 493–504, 2024, doi: 10.1093/jssam/smad015.

[2] O. Galal, A. H. Abdel-Gawad, and M. Farouk, "Rethinking of BERT Sentence Embedding for Text Classification," Neural Comput. Appl., vol. 36, no. 32, pp. 20245–20258, 2024, doi: 10.1007/s00521-024-10212-3.

[3] X. Fu et al., "SS-BERT : A Semantic Information Selecting Approach for Open-Domain Question Answering," MDPI Electron., pp. 1–14, 2023, doi: 10.3390/electronics12071692.

[4] M. Khuntia and D. Gupta, "Indian News Headlines Classification using Word Embedding Techniques and LSTM Model," Procedia Comput. Sci., vol. 218, pp. 899–907, 2022, doi: 10.1016/j.procs.2023.01.070.

[5] S. Al Faraby, A. Romadhony, and Adiwijaya, "Analysis of LLMs for educational question classification and generation," Comput. Educ. Artif. Intell., vol. 7, p. 100298, Dec. 2024, doi: 10.1016/j.caeai.2024.100298.

[6] A. Rita Gonçalves, D. Costa Pinto, H. Gonzalez-Jimenez, M. Dalmoro, and A. S. Mattila, "Me, Myself, and My AI: How artificial intelligence classification failures threaten consumers' self-expression," J. Bus. Res., vol. 186, p. 114974, Jan. 2025, doi: 10.1016/j.jbusres.2024.114974.

[7] M. S. Salim and S. I. Hossain, "An Applied Statistics dataset for human vs AI-generated answer classification," Data Br., vol. 54, p. 110240, 2024, doi: 10.1016/j.dib.2024.110240.

[8] Y. Chae and T. Davidson, "Large Language Models for Text Classification: From Zero-Shot Learning to Instruction-Tuning," Sociol. Methods Res., p. 29, Apr. 2025, doi: 10.1177/00491241251325243.

[9] M. O. Gani et al., "Towards Enhanced Assessment Question Classification: a Study using Machine Learning, Deep Learning, and Generative AI," Conn. Sci., vol. 37, no. 1, 2025, doi: 10.1080/09540091.2024.2445249.

[10] H. Sharma, R. Mathur, T. Chintala, S. Dhanalakshmi, and R. Senthil, "An Effective Deep Learning Pipeline for Improved Question Classification into Bloom's Taxonomy's Domains," Educ. Inf. Technol., vol. 28, no. 5, pp. 5105–5145, 2023, doi: 10.1007/s10639-022-11356-2.

[11] D. Han, T. Tohti, and A. Hamdulla, "Attention-Based Transformer-BiGRU for Question Classification," Inf., vol. 13, no. 5, 2022, doi: 10.3390/info13050214.

[12] S. Aburass, O. Dorgham, and M. A. Rumman, "An Ensemble Approach to Question Classification: Integrating Electra Transformer, GloVe, and LSTM," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 1, pp. 507–514, 2024, doi: 10.14569/IJACSA.2024.0150148.

[13] L. Escouflaire, A. Descampe, and C. Fairon, "Automated text classification of opinion vs. news French press articles. A comparison of transformer and feature-based approaches," Lang. Commun., vol. 99, pp. 129–140, 2024, doi: 10.1016/j.langcom.2024.09.004.

[14] S. Al Faraby, Romadhony, and A. Romadhony, "Educational Question Classification with Pre-trained Language Models," IEEE Xplore Seventh Int. Conf. Informatics Comput., no. 156, pp. 1–6, 2022, doi: 10.1109/ICIC56845.2022.10006957.

[15] G. Xiao, E. Chow, H. Chen, J. Mo, J. Gui, and X. Gong, "Chinese Question Classification in the Low Domain," Fourteenth IEEE Int. Conf. E-bus. Eng., pp. 214–219, 2017, doi: 10.1109/ICEBE.2017.41.

[16] I. V. C. Motta, N. Vuillerme, H. H. Pham, and F. A. P. de Figueiredo, "Machine Learning Techniques for Coffee Classification: a Comprehensive Review of Scientific Research," Artif. Intell. Rev., vol. 58, no. 1, 2025, doi: 10.1007/s10462-024-11004-w.

[17] K. Fujiwara, "Knowledge Distillation with Resampling for Imbalanced data Classification: Enhancing Predictive Performance and Explainability Stability," Results Eng., vol. 24, no. November, p. 103406, 2024, doi: 10.1016/j.rineng.2024.103406.

[18] I. Martinsen, D. Wade, B. Ricaud, and F. Godtliebsen, "The 3-billion Fossil Question: How to Automate Classification of Microfossils," Artif. Intell. Geosci., vol. 5, no. April, p. 100080, 2024, doi: 10.1016/j.aiig.2024.100080.

[19] S. Rizou et al., "Efficient intent classification and entity recognition for university administrative services employing deep learning models," Intell. Syst. with Appl., vol. 19, p. 200247, Sep. 2023, doi: 10.1016/j.iswa.2023.200247.

[20] H. Sobhanam and J. Prakash, "Analysis of fine tuning the hyper parameters in RoBERTa model using genetic algorithm for text classification," Int. J. Inf. Technol., vol. 15, no. 7, pp. 3669–3677, 2023, doi: 10.1007/s41870-023-01395-4.

[21] D. Sarkar, Text Analytics with Python. Berkeley, CA: Apress, p. 674, 2019, doi: 10.1007/978-1-4842-4354-1.

[22] N. Indurkhya and F. J. Damerau, Handbook of Natural Language Processing, 2nd Editio. United States of America: Chapman and Hall/CRC, 2010, doi: 10.1201/9781420085938.

[23] B. Steven, K. Ewan, and L. Edward, Natural Language Processing with Python. United States of America: O'Reilly Media, Inc, p. 504, 2009. [Online]. Available at: https://www.google.co.id/books/edition/Natural_Language_Processing_with_Python/KGIbfiiP1i4C?hl=en&gbpv=0.

[24] B. Srinivasa-desikan, Natural Language Processing and Computational Linguistics, vol. 6, no. 11. UK: Packt Publishing, pp. 1- 388, 2018. [Online]. Available at: https://www.arcjournals.org/pdfs/ijsell/v6-i11/2.pdf.

[25] M. Swamynathan, Mastering Machine Learning with Python in Six Steps, 2nd ed., vol. 19, no. 2. Berkeley, CA: Apress, 2019, doi: 10.1007/978-1-4842-4947-5.

[26] A. A. Khan, "Balanced Split: A new train-test data splitting strategy for imbalanced datasets," arXiv, pp. 1–5, 2022, [Online]. Available at: https://arxiv.org/pdf/2212.11116.

[27] W. Ai, Z. Wang, H. Shao, T. Meng, and K. Li, "A Multi-semantic PassingFramework for Semi-supervised Long Text Classification," Appl. Intell., vol. 53, no. 17, pp. 20174–20190, 2023, doi: 10.1007/s10489-023-04556-x, doi: 10.1007/s10489-023-04556-x.

[28] M. Candito, "Auxiliary Tasks to Boost Biaffine Semantic Dependency Parsing," Trait. Autom. des Langues Nat. TALN 2022 - Actes la 29e Conf. sur le Trait. Autom. des Langues Nat. Conf. Princ., vol. 1, pp. 424–433, 2022, doi: 10.18653/v1/2022.findings-acl.190.

[29] T. T. Goh, N. A. A. Jamaludin, H. Mohamed, M. N. Ismail, and H. Chua, "Semantic Similarity Analysis for Examination Questions Classification Using WordNet," Appl. Sci., vol. 13, no. 14, 2023, doi: 10.3390/app13148323.

[30] B. Li, Y. Fan, Y. Sataer, and Z. Gao, "A Higher-Order Semantic Dependency Parser," ACL, vol. 1, pp. 1–8, 2022, [Online]. Available at: https://arxiv.org/abs/2201.11312.

[31] X. Duan, H. Zan, X. Bai, and C. Zähner, "Reusable Phrase Extraction Based on Syntactic Parsing," 19th Chinese Natl. Conf. Comput. Linguist. CCL 2020, pp. 1166–1171, 2020, doi: 10.1007/978-3-030-63031-7_33.

[32] A. Basirat and J. Nivre, "Syntactic nuclei in dependency parsing - A multilingual exploration," EACL 2021 - 16th Conf. Eur. Chapter Assoc. Comput. Linguist. Proc. Conf., no. 2000, pp. 1376–1387, 2021, doi: 10.18653/v1/2021.eacl-main.117.

[33] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," Proc. ofthe 15th Conf. ofthe Eur. Chapter ofthe Assoc. Comput. Linguist., vol. 2, pp. 427–431, 2017, doi: 10.18653/v1/E17-2068.

[34] O. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv1301.3781v3 [cs.CL] 7 Sep 2013, pp. 1–12, 2013, [Online]. Available at: https://arxiv.org/abs/1301.3781.

[35] T. Zhou, Y. Wang, and X. Zheng, "Chinese Text Classification Method using FastText and Term Frequency-Inverse Document Frequency Optimization," J. Phys. Conf. Ser., vol. 1693, no. 1–7, 2020, doi: 10.1088/1742-6596/1693/1/012121.

[36] H. M. Linh, N. T. M. Huyen, V. X. Luong, N. T. Luong, P. T. Hue, and L. Van Cuong, "VLSP 2020 Shared Task: Universal Dependency Parsing for Vietnamese," Proc. 7th Int. Work. Vietnamese Lang. Speech Process., pp. 77–83, 2020. [Online]. Available at: https://aclanthology.org/2020.vlsp-1.15/.

[37] J. Choi and S. W. Lee, "Improving FastText with Inverse Document Frequency of Subwords," Pattern Recognit. Lett., vol. 133, pp. 165–172, 2020, doi: 10.1016/j.patrec.2020.03.003.

[38] K. Maity, A. Kumar, and S. Saha, "Attention Based BERT-FastText Model for Hate Speech and Offensive Content Identification in English and Hindi Languages," CEUR Workshop Proc., vol. 3159, pp. 182–190, 2021. [Online]. Available at: https://ceur-ws.org/Vol-3159/T1-18.pdf.

[39] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in Pre-training Distributed word Representations," Lr. 2018 - 11th Int. Conf. Lang. Resour. Eval., no. 1, pp. 52–55, 2017. [Online]. Available at: https://arxiv.org/abs/1712.09405.

[40] A. Vaswani et al., "Attention Is All You Need Ashish," 31st Conf. Neural Inf. Process. Syst. (NIPS 2017), vol. 8, no. 1, pp. 8–15, 2017, doi: 10.1109/2943.974352.

[41] A. G. D'Sa, I. Illina, and D. Fohr, "BERT and fastText Embeddings for Automatic Detection of Toxic Speech," Proc. 2020 Int. Multi-Conference Organ. Knowl. Adv. Technol. OCTA 2020, 2020, doi: 10.1109/OCTA49274.2020.9151853.

[42] M. Liang and T. Niu, "Research on Text Classification Techniques Based on Improved TF-IDF Algorithm and LSTM Inputs," Procedia Comput. Sci., vol. 208, pp. 460–470, 2022, doi: 10.1016/j.procs.2022.10.064.

[43] A. L. I. S. Alammary, "Arabic Questions Classification Using Modified TF-IDF," IEEE Access, vol. 9, pp. 95109–95122, 2021, doi: 10.1109/ACCESS.2021.3094115.

[44] I. K. Nti, O. Nyarko-Boateng, and J. Aning, "Performance of Machine Learning Algorithms with Different K Values in K-fold CrossValidation," Int. J. Inf. Technol. Comput. Sci., vol. 13, no. 6, pp. 61–71, Dec. 2021, doi: 10.5815/ijitcs.2021.06.05.

[45] A. Panesar, "Evaluating Machine Learning Models," in Machine Learning and AI for Healthcare, Berkeley, CA: Apress, 2021, pp. 189–205, doi: 10.1007/978-1-4842-6537-6_7.

[46] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," arXiv, pp. 1–49, 2018, [Online]. Available at: https://arxiv.org/abs/1811.12808.

[47] J. Wei, C. Huang, S. Vosoughi, and J. Wei, "What are People Asking About COVID-19? A Question Classification Dataset," Proc. Annu. Meet. Assoc. Comput. Linguist., pp. 1–8, 2020, [Online]. Available at: https://aclanthology.org/2020.nlpcovid19-acl.8/.