FOURTH YEAR B.TECH. PROECT REPORT

A report submitted in partial fulfilment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING

By

BT21CSE194 GYANBARDHAN

Under Supervision of
DR. NISHAT A. ANSARI

DUPLICATE QUESTION DETECTION

Topics in Artificial Intelligence

CSL 421

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,

NAGPUR

(An Institution of National Importance by Act of Parliament)

Survey No.140,141/1 behind Br.Sheshrao Wankhede Shetkari Sahkari SootGirni ,Nagpur, 441108

TABLE OF CONTENTS

1. Introduction

1.1. Introduction

Duplicate questions on online Q&A platforms, such as Stack Overflow and Quora, are a common issue. These duplicates hinder user experience by cluttering the platform with redundant content, making it harder for users to find accurate and relevant answers efficiently. Given the exponential growth of online communities, it has become crucial to implement systems that can identify and flag duplicate or near-duplicate questions. This project leverages natural language processing (NLP) techniques, combined with machine learning models, to address this issue by automating the detection of duplicate question pairs, ultimately streamlining the user experience and improving platform quality.

1.2. Problem Statement

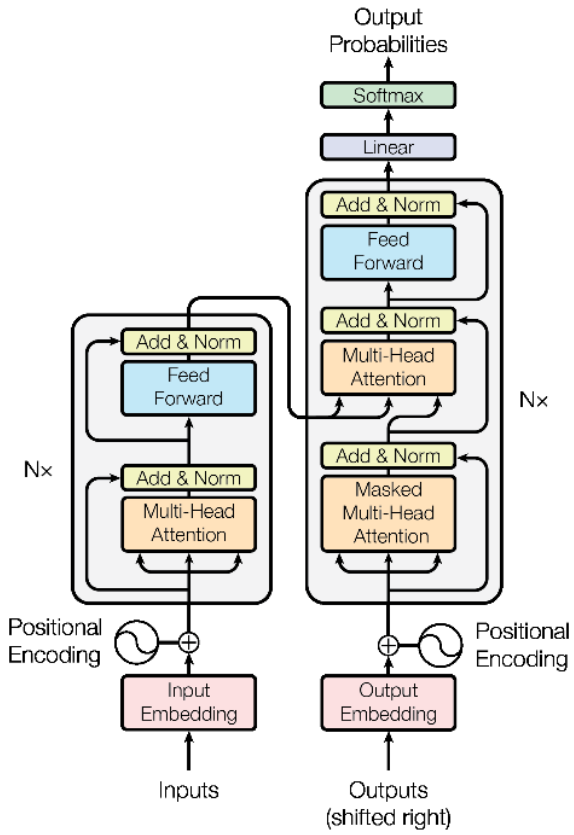
Online question-answer platforms like Stack Overflow, Quora, and Reddit receive a massive volume of user-generated questions on a daily basis. A significant portion of these questions are often repetitive, covering similar topics in various wordings. Such duplicates clutter the platform and degrade user experience by making it difficult for users to quickly find unique and relevant answers. This project seeks to develop a solution that accurately identifies duplicate question pairs using NLP-based models, thus improving the efficiency of search functions and reducing content redundancy on Q&A platforms.

2. Related Studies

2.1 Paper 1

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin



1. **Introduction of the Transformer Model:** Proposed a new architecture for sequence transduction tasks, replacing RNNs and LSTMs.
2. **Core Concept**: Utilized self-attention mechanisms instead of recurrent structures for better parallelization.
3. **Self-Attention Mechanism**:
   - Computes weights for each word in a sequence based on its relation to other words.
   - Captures long-range dependencies effectively.
4. **Scalability and Speed**:
   - Processes entire sequences simultaneously, unlike RNNs which process sequentially.
   - Results in faster training and better use of computational resources.
5. **Positional Encoding**:
   - Introduced to incorporate the order of the sequence, as the self-attention mechanism lacks inherent positional awareness.

6. **Multi-Head Attention**:
   - Uses multiple attention heads to capture information from different representation subspaces.
7. **Layer Structure**:
   - Stacked layers of encoders and decoders to build a deep network.
   - Each layer contains multi-head attention and feed-forward neural networks.
8. **Superior Performance**:
   - Outperformed previous models in tasks like machine translation.
   - Paved the way for advanced models like BERT and GPT.
9. **Impact on NLP**:
   - Revolutionized NLP by providing a foundation for transformer-based architectures.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

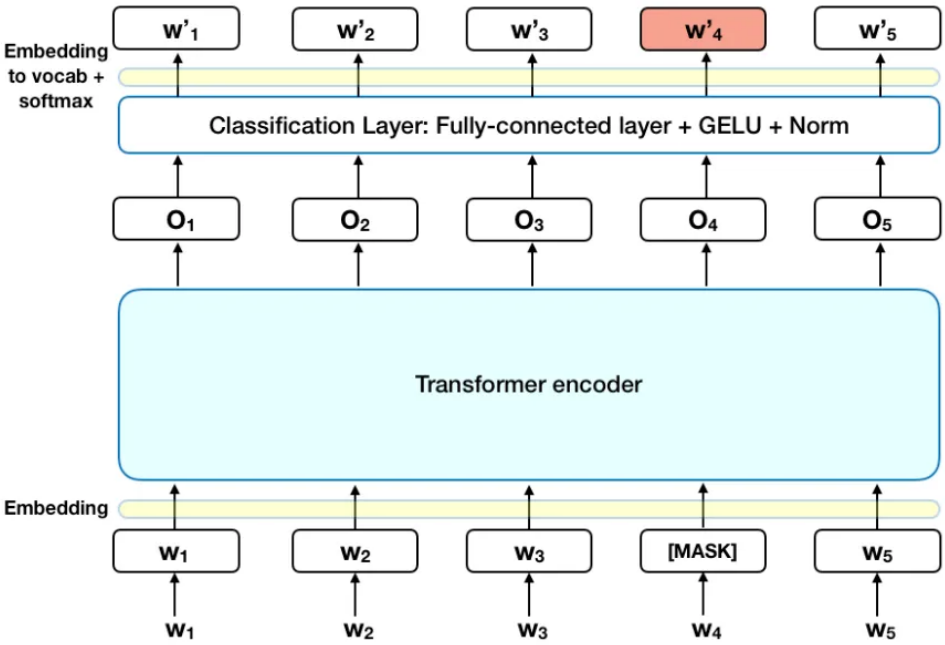
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention. For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

2.2 Paper 2

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova



This paper introduces a new language representation model called **BERT** (Bidirectional Encoder Representations from Transformers). Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It achieves new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to **80.5%** (7.7% absolute improvement), MultiNLI accuracy to **86.7%** (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to **93.2** (1.5-point absolute improvement), and SQuAD v2.0 Test F1 to **83.1** (5.1-point absolute improvement).

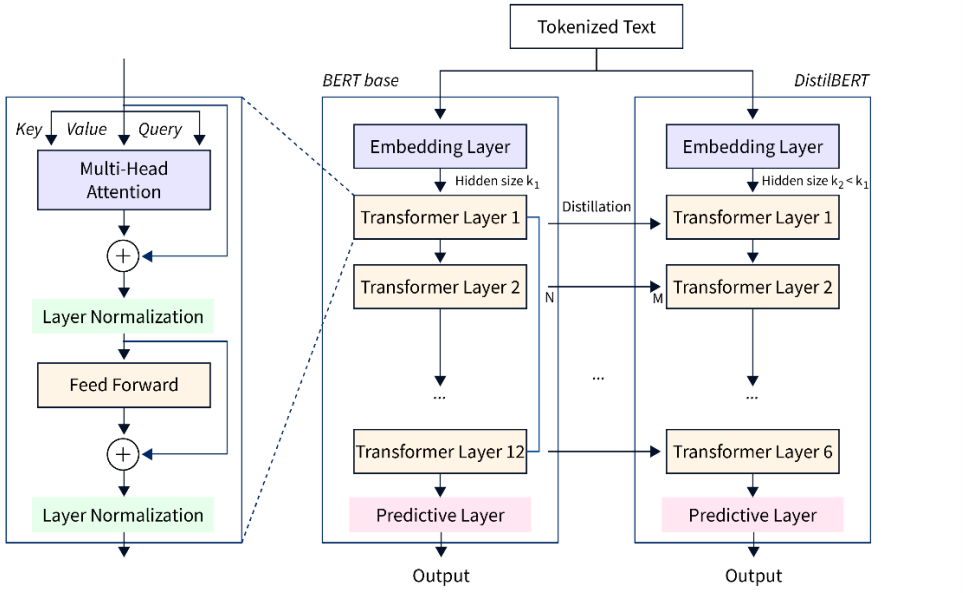
The GLUE test results, evaluated by the official server (https://gluebenchmark.com/leaderboard), report metrics such as F1 scores, Spearman correlations, and accuracy for various tasks. The "Average" excludes the problematic WNLI set, and results are based on single-model, single-task evaluations for BERT and OpenAI GPT.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

2.3 Paper 3

DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf



As transfer learning from large-scale pre-trained models becomes increasingly prevalent in Natural Language Processing (NLP), operating these large models on edge devices and/or under constrained computational training or inference budgets remains a significant challenge. This research proposes a method to pre-train a smaller general-purpose language representation model, called DistilBERT, which achieves competitive performance on a wide range of tasks, comparable to its larger counterparts.
Unlike prior works that focused on task-specific models, this study leverages knowledge distillation during the pre-training phase, demonstrating that it is possible to reduce the size of a BERT model by 40% while retaining

97% of its language understanding capabilities and improving processing speed by 60%. To harness the inductive biases learned by larger models, a triple loss is introduced, combining language modeling, distillation, and cosine-distance losses. This smaller, faster, and more efficient model is cheaper to pre-train and proves its capabilities for on-device computations through a proof-of-concept experiment and comparative on-device study.

DistilBERT retains 97% of BERT performance. Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors . BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Related Work :

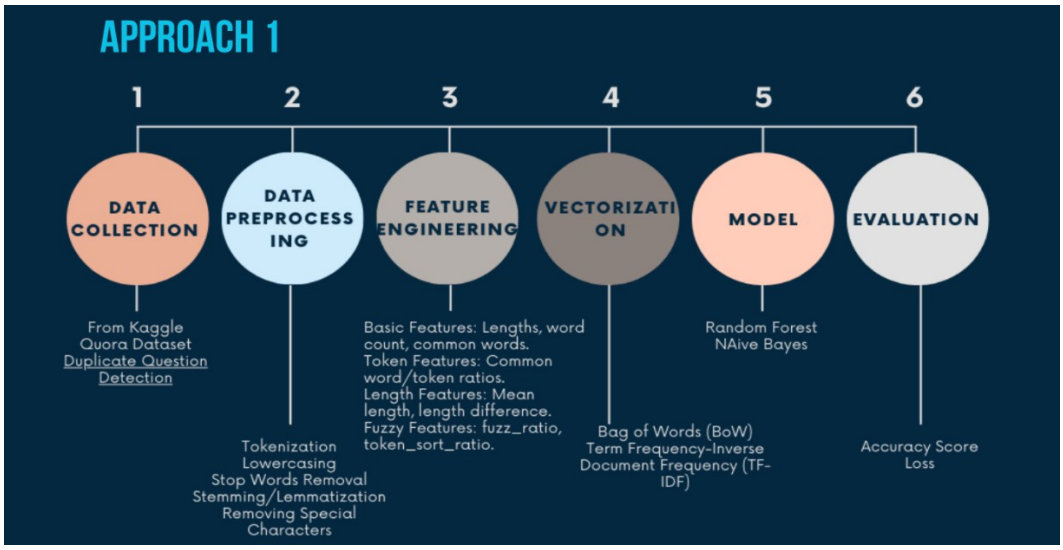
The section explores techniques for compressing large language models, emphasizing task-specific and general-purpose distillation, as well as other methods like pruning and quantization. Task-specific distillation focuses on transferring knowledge from large models like BERT to smaller models tailored for specific tasks. For instance, Tang et al. (2019) distilled BERT into an LSTM classifier, while Chatterjee (2019) distilled a fine-tuned BERT on SQuAD into a smaller transformer model. However, the study highlights that general-purpose pre-training distillation, which leverages the teacher model's knowledge during pre-training, offers broader benefits over task-specific setups, as supported by ablation studies. Multi-distillation techniques, such as those by Yang et al. (2019), combine knowledge from multiple teacher models using multi-task learning, leading to compact question-answering or multilingual models. Additionally, other compression techniques, like pruning (removing attention heads without significant performance loss) and quantization (reducing numerical precision), are complementary but orthogonal to the primary focus on pre-training distillation. These methods collectively demonstrate effective ways to compress models while maintaining performance.

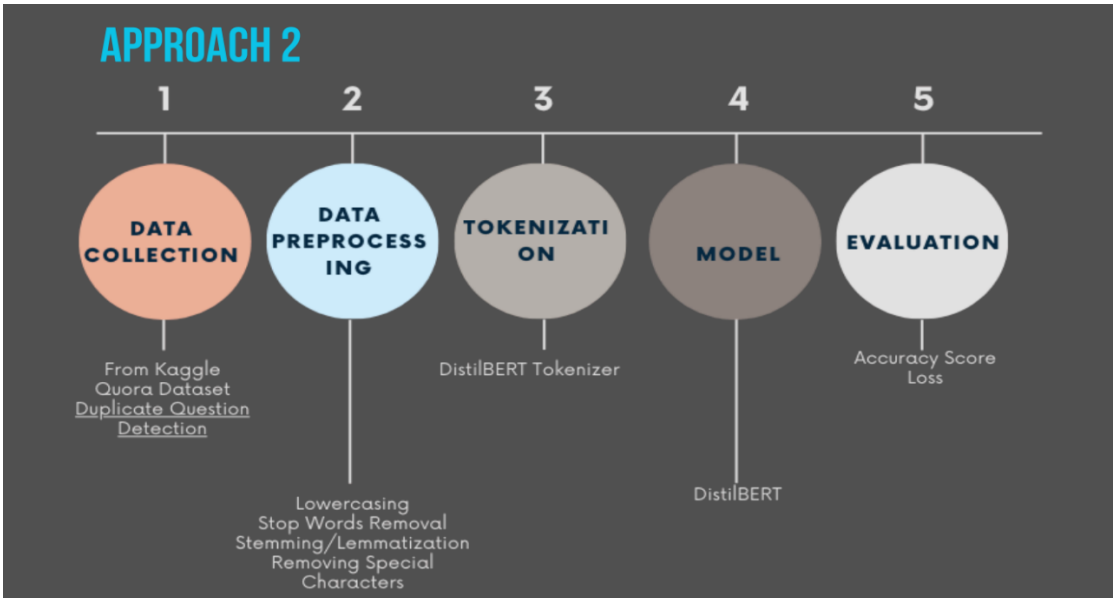
3. Problem Definition

3.1. Problem Statement

Online question-answer platforms like Stack Overflow, Quora, and Reddit receive a massive volume of user-generated questions on a daily basis. A significant portion of these questions are often repetitive, covering similar topics in various wordings. Such duplicates clutter the platform and degrade user experience by making it difficult for users to quickly find unique and relevant answers. This project seeks to develop a solution that accurately identifies duplicate question pairs using NLP-based models, thus improving the efficiency of search functions and reducing content redundancy on Q&A platforms.

3.2. Flowchart/Block diagrams





3.3. Dataset Description

The dataset used in this project is a comprehensive and large collection of question pairs derived from well-known Q&A platforms such as Stack Overflow, Quora, and similar online forums. These question pairs are labeled as either "duplicate" or "non-duplicate," serving as the ground truth for training and evaluating the model. The labels indicate whether two questions are semantically identical or distinct in meaning, enabling the model to learn the nuances of identifying redundant or related content.

- **Total Rows:** 404,290
- **Total Columns:** 6
- **Dataset Source:** Kaggle (https://www.kaggle.com/datasets/gyanbardhan/quora-duplicate-questions-copy)

- **Split Ratios:**
  - **Training Set:** 80% of the data
  - **Validation/Test Set:** 20% of the data

Column Descriptions:

1. **id:** A unique identifier for each row in the dataset.
2. **qid1:** A unique identifier for the first question in the pair.
3. **qid2:** A unique identifier for the second question in the pair.
4. **question1:** The text of the first question in the pair.
5. **question2:** The text of the second question in the pair.
6. **is_duplicate:** A binary label indicating whether the two questions are semantically the same (1 for duplicate, 0 for non-duplicate).

Dataset Splitting Strategy:

The dataset is systematically divided into two subsets to ensure robust training, validation, and evaluation:

- **Training Set:** Used to train the model. Contains 80% of the data (323,432 rows).
- **Validation/Test Set:** Used to tune model hyperparameters and prevent overfitting.Also used for evaluation . Contains 20% of the data (80,858 rows).

This structured approach to dataset division helps ensure that the model is properly trained, validated, and evaluated, leading to a more reliable and effective duplicate question detection system.

4. Model Building

4.1 Approach 1

Six steps:

- data collection (Kaggle Quora Dataset),
- preprocessing (tokenization, lowercasing, stop words removal, stemming),
- feature engineering (lengths, token ratios, fuzzy similarity),
- vectorization (Bag of Words, TF-IDF),
- modeling (Random Forest, Naïve Bayes), and
- evaluation (accuracy, loss) for detecting duplicate questions.

Feature Engineering

Feature engineering involves creating meaningful features from the raw question pairs to help the model differentiate between duplicate and non-duplicate questions. Here's a breakdown of the features and how they are calculated:

I. Basic Features:

1. **q1_len and q2_len:** The character lengths of question1 and question2.
   - Example:
     question1: *"What is AI?"* → q1_len = 11

question2: *"Define artificial intelligence."* → q2_len = 29

2. **q1_words and q2_words:** The number of words in each question.
    - Example:
      question1: *"What is AI?"* → q1_words = 3
      question2: *"Define artificial intelligence."* → q2_words = 3
3. **words_common:** The count of unique words common between the two questions.
    - Example: Common words between *"What is AI?"* and *"Define artificial intelligence."*: [] → words_common = 0
4. **words_total:** The total number of words in both questions combined.
    - Example: words_total = q1_words + q2_words = 3 + 3 = 6
5. **word_share:** Ratio of common words to total words.
    - Formula: word_share = words_common / words_total
    - Example: word_share = 0 / 6 = 0

II. Token Features:

These features analyze the overlap of tokens and stop words between questions. Tokens are meaningful units derived from questions, typically words after preprocessing (e.g., lowercasing, removing punctuation).

1. **cwc_min and cwc_max:** Ratios of common words to the smaller and larger question length.
    - Example:
      question1: *"What is AI?"*
      question2: *"Define artificial intelligence."*
      Common words: 0
      cwc_min = cwc_max = 0
2. **csc_min and csc_max:** Ratios of common stop words (e.g., *is, the*) to smaller/larger stop word counts.
    - Example: Stop words in *"What is AI?"*: [*is*]. csc_min = 1/1 = 1.
3. **last_word_eq and first_word_eq:** Binary features checking if the first or last words are the same.
    - Example: *"What is AI?"* and *"What is machine learning?"*
      first_word_eq = 1, last_word_eq = 0.

III. Length-Based Features:

1. **mean_len:** Average length (word count) of the two questions.
    - Formula: (q1_words + q2_words) / 2
    - Example: mean_len = (3 + 3) / 2 = 3.
2. **abs_len_diff:** Absolute difference in word count between the two questions.
    - Example: *"What is AI?"* (3 words) and *"Define artificial intelligence."* (3 words).
      abs_len_diff = abs(3 - 3) = 0.
3. **longest_substr_ratio:** Ratio of the length of the longest common substring to the smaller question length.
    - Example: Longest common substring: *"is"* (2 characters).
      longest_substr_ratio = 2 / 11 = 0.18.

IV. Fuzzy Features:

Fuzzy matching measures text similarity. Using the fuzzywuzzy library, several similarity scores are computed:

1. **fuzz_ratio:** General similarity score between the two questions.
2. **fuzz_partial_ratio:** Similarity score based on partial matching.
3. **token_sort_ratio:** Similarity score after sorting the tokens in both questions.
4. **token_set_ratio:** Similarity score using set operations to find overlap between token sets.

- Example: *"What is AI?"* and *"Define artificial intelligence."*
    - fuzz_ratio = 35
    - token_set_ratio = 40

Vectorization

Vectorization transforms textual data into numerical representations for machine learning models. Methods like Bag of Words (BoW) and TF-IDF encode text by token frequencies or importance, enabling algorithms to process and analyze semantic relationships.

Bag of Words (BoW) and TF-IDF Features:

- **Bag of Words (BoW):**
  Represents each question as a sparse matrix where rows correspond to questions and columns to unique tokens. The matrix contains token counts for each question.
  Example: *"What is AI?"* → [1, 1, 1, 0, 0, 0].

- **TF-IDF:**
  Similar to BoW but assigns higher weights to rare but important words and reduces weights of frequent ones (e.g., *is, the*).

Model: Random Forest Classifier

The **Random Forest Classifier** is a tree-based ensemble model that uses these engineered features to predict whether a question pair is a duplicate.

How It Works:

1. **Input Features:**
    - Engineered features like q1_len, word_share, fuzz_ratio, cwc_min, etc.
2. **Training:**
    - Each decision tree in the forest is trained on a random subset of features and data samples to learn patterns distinguishing duplicate and non-duplicate pairs.
3. **Prediction:**
    - For a new question pair, each tree predicts a label (duplicate or non-duplicate), and the forest aggregates these predictions (e.g., majority vote).

Example:

Input Features:

- q1_len = 11, q2_len = 29, words_common = 0, word_share = 0, fuzz_ratio = 35.

Prediction:

- The Random Forest Classifier outputs 0 (non-duplicate).

Advantages of Random Forest:

- Handles both numerical and categorical features.
- Resistant to overfitting due to averaging across trees.

- Effective for datasets with engineered features.

4.2 Approach 2

Utilizes the Kaggle Quora Dataset, preprocesses it (lowercasing, stop-word removal, stemming), tokenizes it using the DistilBERT tokenizer, trains a DistilBERT model, and evaluates performance using accuracy and loss metrics.

Tokenization:

DistilBERT's tokenizer splits each question into tokens (subword units) and maps them to unique IDs in its vocabulary. Special tokens like [CLS] (start of input) and [SEP] (separator) are added to mark question boundaries.

- Example:
  Question 1: "What is AI?" → [CLS] What is AI ? [SEP] → [101, 2054, 2003, 993, 1029, 102]
  Question 2: "Explain artificial intelligence." → [CLS] Explain artificial intelligence . [SEP] → [101, 4860, 2450, 3514, 1012, 102]

Feature Extraction:

DistilBERT's transformer layers use **self-attention** to compute the contextual relationships between tokens in both questions. Each token attends to all others in the input sequence, generating context-aware embeddings.

- **What is Self-Attention?**
  Self-attention allows the model to **weigh the importance of each token in the input sequence relative to every other token**. It calculates how much focus each token should have on the others when forming context-aware embeddings. This enables the model to understand relationships between tokens and capture semantic nuances.

  Example Scenario:
  Consider the two questions:
  "What is AI?"
  "Explain artificial intelligence."

  Tokens for the combined input after tokenization and special token addition:
  [CLS] What is AI ? [SEP] Explain artificial intelligence . [SEP]

- **Steps in Self-Attention**

  Input Embeddings:
  Each token is first mapped to a high-dimensional vector embedding from the DistilBERT vocabulary. For instance:
    - *"What"* → [0.45, -0.12, 0.34, ...]
    - *"AI"* → [0.22, 0.88, -0.44, ...]

  Projection to Query, Key, and Value Vectors:
  For every token, the model generates three vectors via learned linear transformations:

- **Query (Q):** Determines which tokens to focus on.
- **Key (K):** Represents the information each token contains.
- **Value (V):** Encodes the content of the token.

Example for "AI":

- *Query Vector: [0.12, 0.08, -0.34, ...]*
- *Key Vector: [-0.22, 0.51, 0.10, ...]*
- *Value Vector: [0.03, 0.90, -0.12, ...]*

Attention Scores (Similarity Calculation):
To determine how much focus *"AI"* should place on other tokens, the dot product between its **Query vector** and the **Key vectors** of all tokens is computed. The scores are scaled and passed through a softmax function to convert them into probabilities (attention weights).

Attention Weight Example:
Token *"AI"* attends more to *"artificial intelligence"* than to *"What"* or *"is"*:

- Attention to *"artificial"* → 0.7
- Attention to *"intelligence"* → 0.6
- Attention to *"What"* → 0.1

Weighted Sum of Values:
The attention weights are multiplied with the **Value vectors** of all tokens, and the weighted sum is calculated to produce the updated representation for *"AI"*.

- *Updated Representation for "AI" → [0.19, 0.85, -0.05, ...]*

- **Capturing Semantic Relationships:**
  The self-attention mechanism enables the model to:
  - Recognize that *"AI"* and *"artificial intelligence"* are semantically related.
  - Ignore irrelevant tokens like *"What"* and *"Explain"* for the specific context of similarity detection.

  This ensures that **DistilBERT can encode both local (within-question) and global (cross-question) relationships**.

- **Multi-Head Attention:**
  DistilBERT employs **multi-head self-attention**, where the process described above is repeated in parallel multiple times with different sets of Query, Key, and Value vectors. Each head learns different types of relationships:
  - One head might focus on synonyms (*"AI"* ↔ *"artificial intelligence"*).
  - Another head might focus on sentence structure (*"What is"* ↔ *"Explain"*).

  The outputs of all heads are concatenated and passed through another linear layer to produce the final contextualized embeddings

  Using the self-attention mechanism:
  - The model learns that *"AI"* in the first question corresponds strongly to *"artificial intelligence"* in the second question.
  - These contextual embeddings are passed through a classification head to predict the label (duplicate or non-duplicate).

- **Example:**
  Input:
  - Question 1: *"What is AI?"*
  - Question 2: *"Explain artificial intelligence."*

  Attention Output:
  - Strong relationships between *"AI"* ↔ *"artificial intelligence"* result in a high similarity score.

  Prediction:
  - Label: 1 (Duplicate).

Training:

The output embeddings from DistilBERT are fed into a classification head. The model predicts whether a question pair is a duplicate (1) or non-duplicate (0).
Example:

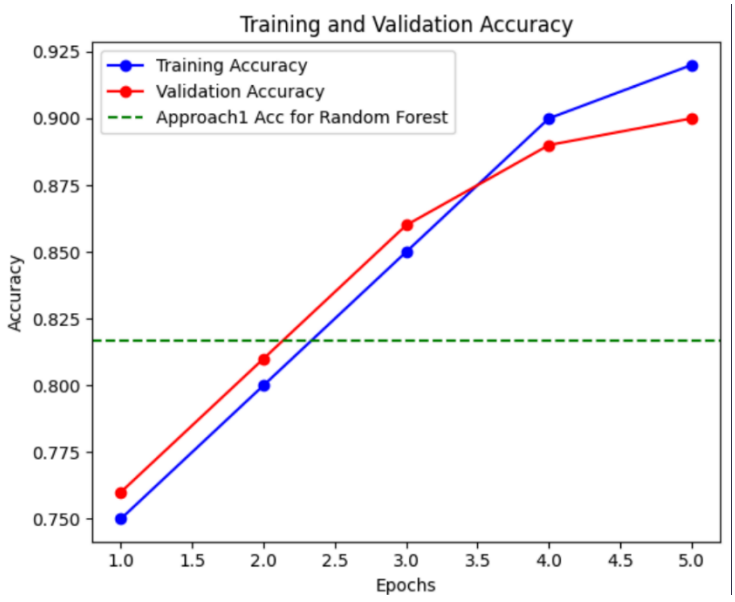
- *Input Pair:*
  Question 1: "What is AI?"
  Question 2: "Explain artificial intelligence."
- *Predicted Label*: 1 (Duplicate)

During training, the model minimizes the cross-entropy loss between its predictions and the ground truth labels using the labeled dataset.

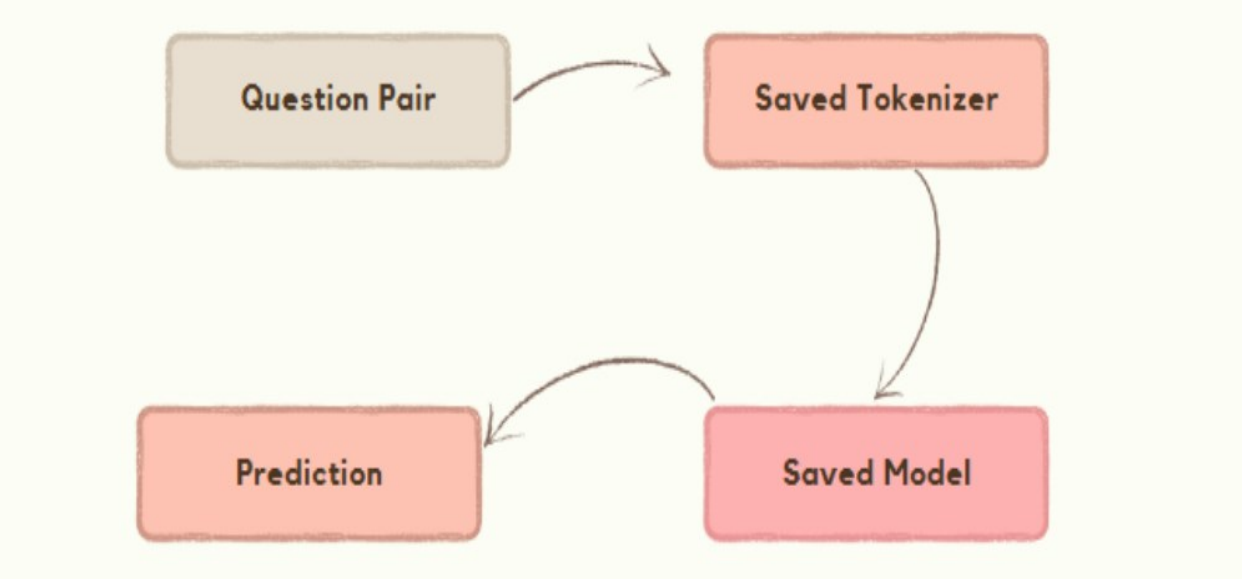
By leveraging tokenization, self-attention, and pre-trained embeddings, DistilBERT effectively captures the semantic relationships between questions, making it powerful for duplicate question detection.

5. Results and discussion

- ## Performance metrics



- **Results and explanation:**




```
question_pairs = [
    ("How do I reset my password?", "What is the process to reset my password?"),
    ("Where can I find my order history?", "How do I view my past orders?"),
    ("What is the return policy for online purchases?", "How can I return an item bought online?"),
    ("Can I change my delivery address after ordering?", "Is it possible to update the shipping address once the order is
    ("How do I contact customer support?", "What's the best way to reach customer service?"),
    ("How do I reset my password?", "How do I delete my account?"),
    ("Where can I find my order history?", "What are the delivery options for my area?"),
    ("What is the return policy for online purchases?", "How long does it take to get a refund after returning an item?"),
    ("Can I change my delivery address after ordering?", "How do I apply a discount code to my order?"),
    ("How do I contact customer support?", "What payment methods are accepted?")
]
```

```
inputs = tokenizer(
    [q[0] for q in question_pairs], [q[1] for q in question_pairs],
    return_tensors='tf',
    truncation=True,
    padding=True,
    max_length=50
)
```

```
outputs = model(inputs)
logits = outputs.logits

# Convert logits to probabilities
probabilities = tf.nn.softmax(logits, axis=-1)
predictions = tf.argmax(probabilities, axis=1).numpy()  # 0 or 1 for binary classification
```

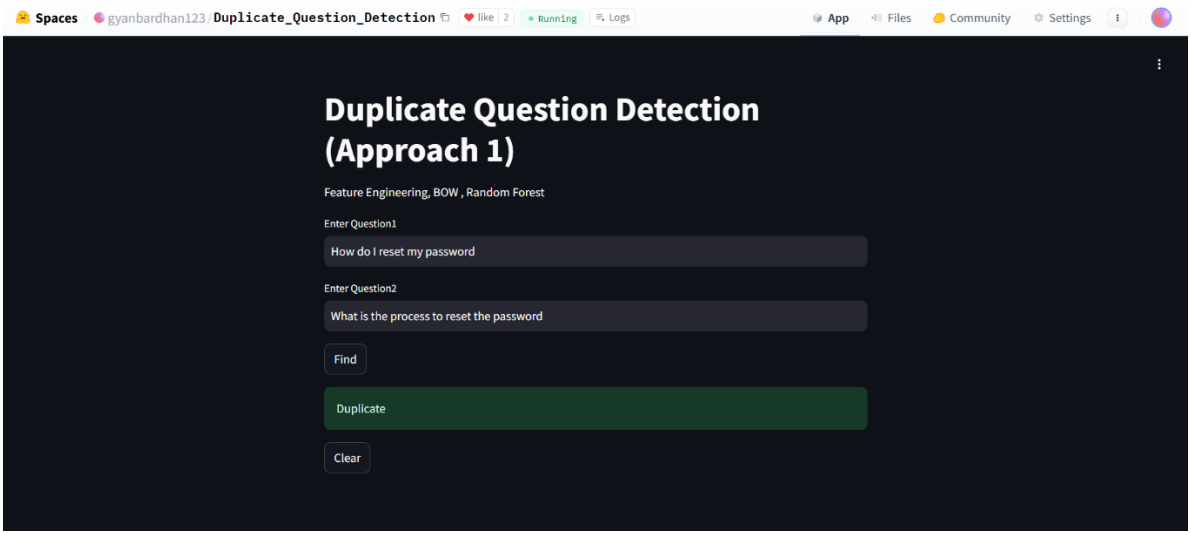
```
Question 1: How do I reset my password?
Question 2: What is the process to reset my password?
Prediction: Duplicate
Probability: [0.12865898 0.871341  ]
------
Question 1: Where can I find my order history?
Question 2: How do I view my past orders?
Prediction: Not Duplicate
Probability: [9.9971086e-01 2.8908864e-04]
------
Question 1: What is the return policy for online purchases?
Question 2: How can I return an item bought online?
Prediction: Not Duplicate
Probability: [9.9984324e-01 1.5668685e-04]
------
Question 1: Can I change my delivery address after ordering?
Question 2: Is it possible to update the shipping address once the order is placed
Prediction: Not Duplicate
Probability: [0.9857459  0.01425405]
------
```

6. Deployment

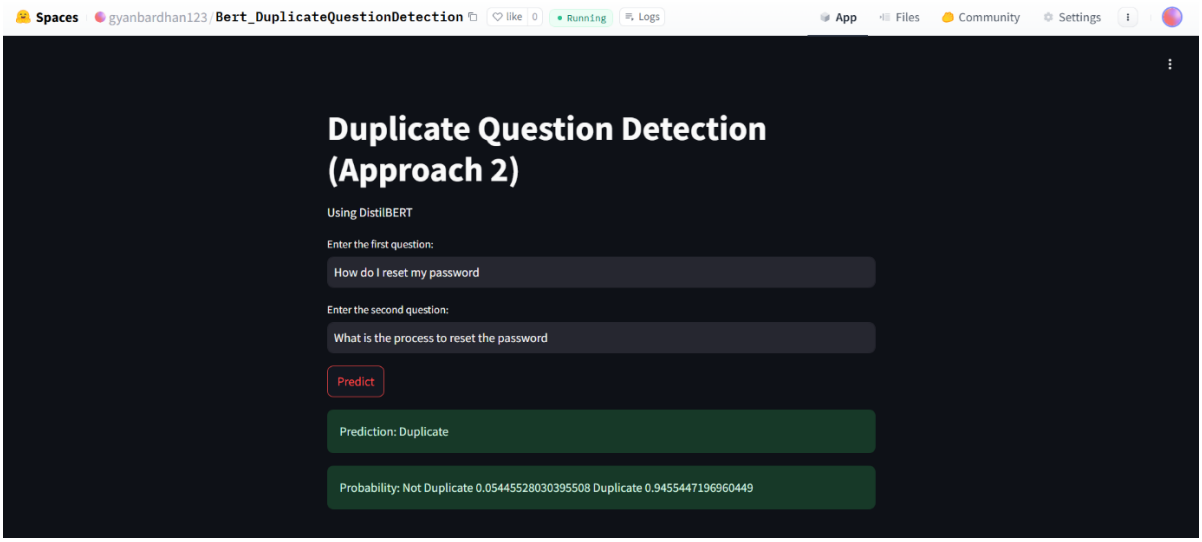
Hugging Face Spaces:

Both approaches are deployed on Hugging Face Spaces using a 16 GB CPU environment.

- **Approach 1**, vectorization techniques like TF-IDF and models such as Random Forest are implemented using scikit-learn.
    - https://huggingface.co/spaces/gyanbardhan123/Duplicate_Question_Detection



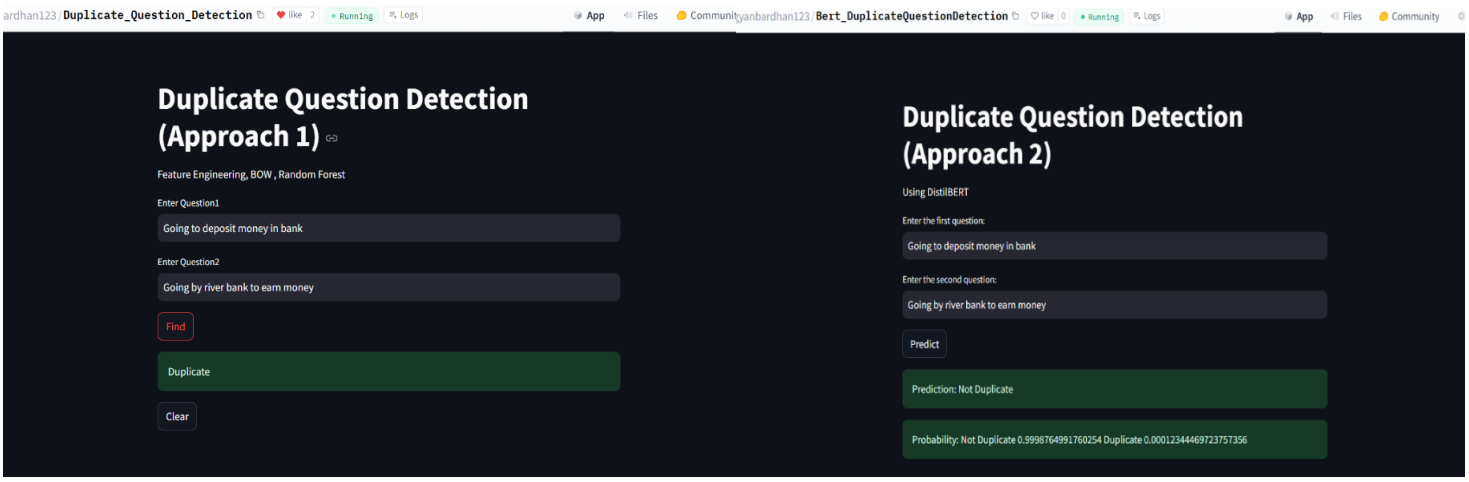
- **Approach 2**, DistilBERT from Hugging Face's transformers library is used for tokenization and model training.
    - https://huggingface.co/spaces/gyanbardhan123/Bert_DuplicateQuestionDetection



Deployment leverages lightweight libraries optimized for CPU, ensuring efficient processing and evaluation.

7. Conclusion

- This project addresses the issue of duplicate questions on online platforms by using natural language processing and machine learning to quickly identify and organize similar questions. This improves platform efficiency, reduces repetitive content, and helps users find answers more easily.
- We compared two models for detecting duplicate questions:
    - BOW/TF-IDF with Random Forest Classifier: Achieved 81.67% accuracy but lacks a deep understanding of context.
    - DistilBERT Transformer: Reached a higher 89.89% accuracy, capturing better context and meaning in questions.



- The provided example demonstrates the difference in contextual understanding between the two approaches:
    - **Question 1:** *"Going to deposit money in bank"*
    - **Question 2:** *"Going by river bank to earn money"*

❖ **Approach 1 (BOW with Random Forest):**
This method labels the two questions as **Duplicate**. This is because it relies on surface-level features, such as common words ("bank" and "going"), without understanding their context. Here, "bank" refers to two different meanings (financial institution vs. riverbank), but BOW treats them as the same.

❖ **Approach 2 (DistilBERT):**
This method correctly identifies the two questions as **Not Duplicate**. DistilBERT uses contextual embeddings to understand that "bank" has different meanings in the two questions and recognizes that their intents (depositing money vs. earning money by a river) are unrelated. Its semantic understanding prevents misclassification.

- DistilBERT outperformed the traditional model by over 8%, making it the preferred choice due to its ability to understand question context more effectively, resulting in a more accurate and reliable solution

7. References

[1]. Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, Polosukhin, Illia. "Attention Is All You Need." arXiv, vol. 1706.03762, 12 Jun. 2017, revised 2 Aug. 2023, https://doi.org/10.48550/arXiv.1706.03762.

[2]. Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, Toutanova, Kristina. "BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding." arXiv, vol. 1810.04805, 11 Oct. 2018, revised 24 May 2019, https://doi.org/10.48550/arXiv.1810.04805.

[3]. Sanh, Victor, Debut, Lysandre, Chaumond, Julien, Wolf, Thomas. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv, vol. 1910.01108, 2 Oct. 2019, revised 1 Mar. 2020, https://doi.org/10.48550/arXiv.1910.01108

8. Important Links

- Web Application –
  Approach 1 :
  https://huggingface.co/spaces/gyanbardhan123/Duplicate_Question_Detection
  Approach 2 :
  https://huggingface.co/spaces/gyanbardhan123/Bert_DuplicateQuestionDetection

- Notebooks-
  Approach 1 :
  https://huggingface.co/spaces/gyanbardhan123/Duplicate_Question_Detection/blob/main/bow-00.ipynb
  https://huggingface.co/spaces/gyanbardhan123/Duplicate_Question_Detection/blob/main/TF-IDF.ipynb
  Approach 2 :
  https://huggingface.co/spaces/gyanbardhan123/Bert_DuplicateQuestionDetection/blob/main/Bert%20Duplicate%20Question%20Detection.ipynb

- Final Dataset- https://www.kaggle.com/datasets/gyanbardhan/quora-duplicate-questions-copy