References Validation in Scholarly Articles using RoBERTa

by

Abdullah Umar Nasib
19166009

This project submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Engg. in Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
May 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Abdullah Umar Nasib
19166009
abdullah.umar.nasib@g.bracu.ac.bd

Approval

The project titled "References Validation in Scholarly Articles using RoBERTa" submitted by Abdullah Umar Nasib (19166009)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Engg. in Computer Science on May 25, 2023.

Examining Committee:

Supervisor:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
rabiul.alam@bracu.ac.bd

Program Coordinator:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson, Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Abstract

In the recent era of technological advancement, evaluating references and assignments and validating those are some of the primary processes to justify the authenticity of a research work in the academic sector. The purpose of referencing is to ensure ethical writing and to make the search easier in a particular area for the reader when it is accurate. The traditional way to cross-check the validations is to check them manually one by one, which is above argument, equivalent to another work, and sometimes so tiring that the reader loses interest in reading the original paper. The target of this research paper is to introduce a semi-automatic digital system that enables researchers to justify the references used in a research paper without doing it manually. In this model, we have prepared a sentence transformer named RoBERTa for generating embedding. We sanitize and pre-process an entire research paper and cross-match that against a reference query using the proposed model in terms of finding semantic and contextual similarity. The result shows mostly similar contexts based on the similarity check. We have compared the embedding of query and user input articles with the help of K similar search function. Our model outperformed the existing BERT and SBERT models' output in accuracy with a F1 score of 0.777, which establishes the fact that the model can be used in real life with a simple query of text from research articles.

Keywords: Reference Validation, Semantic Similarity Analysis, Contextual Similarity Analysis, RoBERTa, K Similar Search, Natural Language Processing

Acknowledgement

I express my gratitude to almighty Allah for allowing me to successfully complete the semester both physically and financially without facing any major illnesses or emergencies.

I would also like to extend my sincere thanks to my esteemed supervisor, **Dr. Md. Golam Rabiul Alam**, Professor of the Department of Computer Science and Engineering at Brac University, for his guidance and support throughout this project and the writing of this paper. His constant supervision and advice kept me on track, and I am truly grateful for having him as my supervisor.

I would like to acknowledge the unwavering support of my parents, as well as my wife, who provided tremendous assistance during every stage of my master's work.

Lastly, I would like to express my appreciation to Brac University for providing me with the necessary resources and support to conduct this research.

Table of Contents

Declaration **i**

Approval **ii**

Abstract **iii**

Acknowledgment **iv**

Table of Contents **v**

List of Figures **vii**

List of Tables **viii**

Nomenclature **ix**

1 Introduction **1**
- 1.1 Motivation 2
- 1.2 Problem Statement 2
- 1.3 Contribution 3
- 1.4 Organization of the Report 4

2 Related Work **5**

3 Dataset **9**
- 3.1 Dataset Preparation 10
- 3.2 Dataset Description 11

4 Methodology **13**
- 4.1 Proposed Model 13
  - 4.1.1 Model Inputs 14
  - 4.1.2 Input Preprocessing 15
  - 4.1.3 RoBERTa Tokenization 16
  - 4.1.4 RoBERTa Sentence Embeddings 17
  - 4.1.5 RoBERTa Vector Generation 19
  - 4.1.6 Computing Semantic Textual Similarity 20
  - 4.1.7 Model Outputs 21
- 4.2 RoBERTa 22
  - 4.2.1 RoBERTa Architecture 22
  - 4.2.2 Mathematically RoBERTa 23

4.3 K Similarity 24

5 Implementation and Result Analysis 27

5.1 Implementation Details 27

5.2 Experimental Setup 27

5.2.1 Similarity check with Query: 28

5.3 Result Analysis 28

5.3.1 Training Model on Dataset 29

5.3.2 Testing Model on Dataset 31

6 Conclusion 33

6.1 Conclusion 33

6.2 Difficulties to Overcome 33

6.2.1 Limitations 34

6.3 Future Work 34

Bibliography 37

List of Figures

3.1 Sample Dataset 10

4.1 Proposed Workflow for Reference Validation 14
4.2 Input Text Tokenization Workflow 17
4.3 Sentence Embeddings Workflow for RoBERTa 18
4.4 Semantic Textual Similarity Computational Workflow 20
4.5 RoBERTa Input and Output Representation 22

5.1 System Training and Testing Workflow 29
5.2 K Similar Score on Training Data 30
5.3 Test Result Against Dataset for the Model 31

List of Tables

5.1 Three most similar sentences to Query with respective scores 28
5.2 K Similar Scores from Dataset 30
5.3 Test Classification Report 31

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte Pair Encoding
CNN	Convolutional Neural Network
KNN	K Nearest Neighbor
LSA	Latent Semantic Analysis
LSTM	Long Short Term Memory
MLM	Masked Language Modeling
NLI	Natural Language Inference
NLP	Natural Language Processing
NSP	Next Sentence Prediction
OOV	Out Of Vocabulary
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT Pretraining Approach
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RWR	Radar Warning Receiver
SBERT	Siamese Bidirectional Encoder Representations from Transformers
SMFCNN	Single Layer Multi Size Filters Convolutional Neural Network
STS	Semantic Textual Similarity
WoS	Web of Science

Chapter 1

Introduction

Over the past two decades, the landscape of scientific research publishing has undergone a significant transformation, shifting from traditional print publications to online platforms [11]. In this new era, the responsibility of ensuring the quality of research papers and the accuracy of references falls on the scientific writers. However, relying solely on the authors is not sufficient when it comes to processing and validating manuscripts. To minimize reference errors, it is crucial for writers, peer reviewers, editors, and publishers to collaborate throughout the publication process. The evaluation and validation of references are essential steps in verifying the authenticity of research work within the academic sector. Traditionally, this process involved manually cross-checking references one by one, which is not only time-consuming but also tiresome for readers, often causing them to lose interest in the original paper. Moreover, this approach is lengthy, labor-intensive, and expensive, while also increasing the reliance on peer reviewers. The aim of this research paper is to introduce a semi-automatic digital system that enables researchers to validate the references used in their papers without the need for manual verification. By leveraging the RoBERTa model and employing a K Similar method to assess similarity, the proposed framework achieves an accuracy of 77% for the given dataset. The adoption of a semi-automatic digital system offers several advantages over the traditional manual approach. Firstly, it streamlines the reference validation process, saving researchers significant time and effort. Instead of manually cross-checking each reference, the system employs sophisticated algorithms to assess the similarity between the provided references and existing works. This automated process not only enhances efficiency but also reduces the risk of human error. Additionally, the introduction of a digital system reduces the dependency on peer reviewers. While peer review remains a critical aspect of the research publication process, the burden on reviewers is lightened by the implementation of an automated system that can handle the initial validation of references. This allows peer reviewers to focus on other crucial aspects of the paper, such as evaluating the methodology, analyzing the findings, and providing insightful feedback. The accuracy achieved by the proposed framework, 77%, showcases its potential as an effective tool for reference validation. However, it is important to note that this system should be seen as a complement to the existing peer review process rather than a replacement. The human expertise and critical evaluation provided by peer reviewers are still invaluable in ensuring the overall quality of a research paper. The shift towards online scientific research publishing has necessitated new approaches to maintain the quality and authenticity of research papers. The development of a semi-automatic digital system for reference validation presents a promising solution to mitigate errors and streamline the publication process. By leveraging advanced models such as RoBERTa and implementing intelligent algorithms, researchers can effectively justify the references used in their work, enhancing the overall credibility and trustworthiness of scientific publications in the digital age.

1.1 Motivation

Our motivation for conducting research on reference validation came from the desire to improve the accuracy and reliability of research findings in reference validation field. Accurate referencing is critical for ensuring that claims and arguments are supported by credible sources, and that the work of previous researchers is properly acknowledged.

However, the process of reference validation can be time-consuming and challenging, particularly when dealing with a large number of sources or complex information. By conducting research on reference validation, we will be able to identify new and more effective methods for verifying references, which could ultimately improve the accuracy and reliability of research in your field. This could have practical applications in a variety of areas, such as academia, journalism, and law, where accurate referencing is critical.

In addition, by contributing to the body of knowledge on reference validation, we might be able to establish ourselves as an expert in text processing field, and potentially make a meaningful contribution to the broader academic community.

Furthermore, our research could help to fill a gap in the literature on this topic, and potentially inspire further research and development in the field. Ultimately, our motivation was to build a system that reads contents from that x reference and searches for the query (let's say, Q) specified and produces a score. The higher value of the score indicates that it is more likely that there's a sentence with semantic similarity to that of Q in the document.

1.2 Problem Statement

There are several challenges in working with text classifications. Text data can be ambiguous, and it can be difficult to determine the intended meaning of a sentence or phrase. This can lead to misclassification or inaccurate analysis. The meaning of a text can often depend on the context in which it is used. For example, the word "bat" can refer to an animal or a piece of sports equipment. Without proper context, it can be difficult to accurately classify text. In addition, Text data can be voluminous, which can make it difficult to process and analyze. This can also lead to issues with data quality, such as incomplete or irrelevant data. Furthermore, Text data often requires extensive cleaning and preprocessing to remove noise, such as punctuation, stop words, and special characters. This can be a time-consuming process that requires expertise in natural language processing. Text classification problems can sometimes have imbalanced classes, meaning that there may be far more examples of one class than another. This can lead to bias and inaccuracies in the classification model. Choosing the right features to use in text classification can be challenging. This is especially true when dealing with large volumes of text data. Choosing the right features can have a big impact on the accuracy of the classification model. Text classification can become more challenging when working with multilingual data. Different languages have different grammatical rules and nuances that can make classification more complex. Overall, working with text classifications requires specialized knowledge and expertise in natural language processing and machine learning, as well as careful consideration of the unique challenges posed by text data.

1.3 Contribution

The aim of this research is to develop an automated tool to validate references used in research papers during the review process of scholarly articles. The primary objective is to simplify the review process for reviewers by providing an efficient and accurate reference validation system. However, due to the unavailability of a suitable dataset, it was necessary to create one manually by reading through various relevant scholarly articles and cross-checking the references used to determine their relevance. The development of an automated reference validation tool has the potential to significantly improve the efficiency and accuracy of the peer-review process for scholarly articles. Currently, the review process is often time-consuming and labor-intensive, with reviewers having to manually verify the accuracy and relevance of each reference used in a research paper. This can be a daunting task, especially when dealing with large volumes of scholarly articles.

To address this challenge, the proposed tool aims to automate the reference validation process by using natural language processing (NLP) techniques to analyze the text of scholarly articles and determine the relevance of each reference used. This would allow reviewers to quickly and easily assess the accuracy and validity of the references used in a research paper, saving time and effort. However, the lack of an appropriate dataset posed a significant challenge in the development of the tool. To overcome this issue, a manual dataset was prepared by reading through various scholarly articles and cross-checking the references used to determine their relevance. This was a time-consuming process that required a high level of expertise in natural language processing and scholarly research.

To summarize the contribution, it can be described as listed:

1. A semi-automated model that can validate references in scholarly articles and shows the finding based on the analysis to see if the reference is relevant against the scholarly publication or not.

2. The framework also displays the most similar findings that the reference states in an articles. An reviewer can determine the number of similar finding along with the similarity score as s/he wants.

3. Introducing a concept of RoBERTa to process large corpora and capture contexts that are spread over multiple sentences in a PDF document or text document.
4. Enhancing existing pre trained RoBERTa model by adding a mean pooling layer followed by another dense layer to the pooled output with an activation function to reduce the size of embedding vector generated by RoBERTa without sacrificing much of the accuracy.
5. Manually created dataset by reading and cross verifying the used references in over 60 scholarly articles and made the dataset accessible to anyone.
6. Calculating the similarity function using K Similar methods and displaying the accuracy functions of this framework

1.4 Organization of the Report

The second chapter of this paper presents a comprehensive review of the relevant literature pertaining to the techniques and algorithms employed in the project. Various prior research studies are discussed, which have implemented similar methods to those used in this project.

The third chapter outlines the proposed model in detail, providing specific information on how it is implemented. This section delves into the technical aspects of the model, detailing the algorithms, tools, and technologies used to develop it.

In chapter 4, the experimental results obtained from collected data sets are presented and compared in terms of their performance. This section provides an in-depth analysis of the results obtained, presenting graphs, tables, and other visual aids to help readers understand the findings. The section also discusses the strengths and weaknesses of the proposed model, highlighting any areas where it may be improved.

Chapter 5 provides concluding remarks on the paper, summarizing the key findings of the study. This section discusses any difficulties or limitations encountered during the research process and provides recommendations for future work on this system. The section emphasizes the importance of continuing research in this area to further refine the proposed model and improve its performance.

In summary, this paper provides a comprehensive analysis of a proposed model for a specific application. The paper reviews relevant literature, outlines the proposed model, presents experimental results, and concludes with suggestions for future work.

Chapter 2

Related Work

References are symbols that let writers recognise their own thoughts and works by pointing to key literary sources. Correct reference ensures moral writing and facilitates finding pertinent scientific sources. However, there has not been any significant research that aims to contextually validate the allusions.

Barroga et al.[3] highlights the importance of correct referencing as well as the responsibilities of each party engaged in the process of publishing a research paper. To provide excellent reference validation, training on digital bibliography management were prioritized. Overall, this essay describes an antiquated manual method for correctly citing a source while writing a research paper. Ruths et. al. [1] described an automated approach that considered a person's name and their publications. It then calculated that person's publication citation record using a collection of domain-specific vocabulary from the aforementioned sources. It requires three inputs: the author's name, the time period during which his articles were published, and a URL that either contains a list of the author's publications or a pretty thorough description of the author's research. The result of the procedure is a list of publications (including citation counts, if available). If specified, other citation metrics such as h- and e-indices are also returned in the result.

The authors of [2] examine whether CiteULike and Mendeley are helpful for assessing academic influence using a sample of 1613 papers published in Nature and Science in 2007. Online reference managers included CiteULike and Mendeley. Websites known as online reference managers enable users to download, read, and share reference material in online reference libraries. The readership size is displayed in an online reference manager by the users who have saved a certain document, in number. The authors of this research referred to the number of readers as the paper's users. Finding a relationship between Mendeley/CiteULike user counts and WoS (Web of Science) citations was the main objective of this study. In fact, a link was discovered using a sample of 1613 publications that were published in Nature and Science in 2007. The number of users of these online reference management systems, according to the authors, appears to be too low to represent a significant threat to conventional citation indexes, nevertheless.

The authors investigated the reliability of citing in academic papers, the author examined 450 references from ten biomedical journals [4]. A total of 30 articles were picked. 15 references from each article were randomly chosen. 81 of these 450 references were incorrect after human review. These errors were divided into two categories: severe errors and minor errors. 48 minor and 33 serious faults were discovered out of the 81 total errors. In another research paper [13], the authors suggested a Target-Dependent BERT (TD-BERT) paradigm that comes in different forms. Our TD-BERT models frequently outperform others, including more modern BERT-based models, in the classification accuracy of aspect term polarity on the SemEval-2014 and a Twitter dataset. The findings demonstrate that complicated neural networks that previously provided strong results with embedding do not fit well with BERT, while the addition of target information to BERT leads in a steady performance improvement.

In a study conducted by Muhammad et al., the problem of imbalanced datasets was addressed using a single-layer multi-size filters convolutional neural network (SMFCNN). The researchers aimed to evaluate the performance of SMFCNN against sixteen machine learning (ML) baseline models. Additionally, they investigated the impact of preprocessing techniques on the performance of SMFCNN. To assess the effectiveness of SMFCNN, experiments were conducted on three imbalanced datasets of varying sizes. Remarkably, SMFCNN outperformed the baseline classifiers on all three datasets. On the medium-sized dataset, SMFCNN achieved an impressive accuracy score of 95.4%. Similarly, on the big and small size datasets, SMFCNN obtained accuracy scores of 91.8% and 93.3% respectively. The findings of this study highlight the superiority of SMFCNN in handling imbalanced datasets. By utilizing a single-layer multi-size filters convolutional neural network, the model demonstrates its ability to effectively learn and classify imbalanced data. The results obtained by SMFCNN surpass those of the baseline classifiers, indicating its potential for addressing imbalanced classification tasks. In summary, Muhammad et al. [17] conducted a comparative study to evaluate the performance of SMFCNN against several ML baseline models on imbalanced datasets. The results clearly demonstrate the superiority of SMFCNN, with the model consistently outperforming the baseline classifiers. Moreover, the study explores the impact of preprocessing techniques on the performance of SMFCNN, emphasizing the significance of data preprocessing in addressing imbalanced classification tasks. These findings contribute to the body of knowledge in the field of imbalanced dataset classification and provide valuable insights for researchers and practitioners working in this domain.

Existing approaches to learning word vector space representations have been successful in capturing intricate semantic and syntactic patterns using vector arithmetic, yet the underlying mechanism remains unclear [5]. The researchers conducted an analysis to elucidate the necessary model properties for such patterns to emerge in word vectors. Their findings led them to develop a novel global log-bilinear regression model that combines the advantages of two prominent model types in the literature: global matrix factorization and local context window methods. This model effectively utilizes statistical information by training only on non-zero elements in a word co-occurrence matrix, rather than the entire sparse matrix or individual context windows in a large corpus. The resulting vector space demonstrates significant substructure, as demonstrated by its 75% success rate on a recent word analogy task. Additionally, their model outperforms comparable models on similarity tasks and named entity recognition.

The assessment of neural network models for sentence comprehension can be easily conducted through natural language inference (NLI), which measures how well the models capture the full meaning of natural language sentences [10]. Although existing NLI datasets such as SNLI have led to significant improvements in modeling, they are limited in their capacity to cover the full range of English language meanings due to their narrow scope. In this paper, the authors present MultiNLI, a new dataset that offers greater linguistic diversity and complexity and serves as a benchmark for cross-genre domain adaptation. MultiNLI includes text and speech samples from ten different genres, compared to SNLI's limited scope of simple image captions. It also contains a higher percentage of sentences with difficult linguistic phenomena, resulting in lower baseline model performance on MultiNLI than on SNLI. MultiNLI's greater diversity and complexity make it an ideal resource for future research in sentence understanding. Since its initial release in draft form in early 2017, other researchers have found that NLI can be a valuable source task for pre-training and transfer learning in sentence-to-vector models, with SNLI and MultiNLI-trained models outperforming prior models on established transfer learning benchmarks. The authors anticipate that MultiNLI will remain a useful resource for the development and evaluation of sentence understanding methods for many years to come.

The need for extracting useful knowledge from a large amount of textual data in a reasonable time frame has become increasingly important. This study, cited as [18], provides an overview of recent abstractive text summarization techniques using deep learning models. It also presents an analysis of the datasets utilized for training and validating these approaches, their limitations, and features. For single-sentence summary approaches, the Gigaword dataset is commonly used, while the CNN/Daily Mail dataset is used for multi-sentence summary approaches. Metrics commonly applied to evaluate summarization quality are Recall-Oriented Understudy for Gisting Evaluation 1 (ROUGE1), ROUGE2, and ROUGE-L. Challenges faced during the summarization process and their solutions are also investigated in this paper. Recurrent neural networks with an attention mechanism and LSTM are the most popular techniques used for abstractive text summarization. Experimental results indicate that pre-trained encoder models achieve the highest values for ROUGE1, ROUGE2, and ROUGE-L (43.85, 20.34, and 39.9, respectively). Text summarization models encounter challenges during testing, including the lack of a definitive reference token, the presence of out-of-vocabulary (OOV) words, and issues like sentence repetition, inaccuracies, and the potential for false information. These obstacles hinder the models' ability to produce optimal summaries, and further research is needed to address these limitations. These challenges will contribute to the development of more accurate and reliable abstractive text summarization systems.

Another study [15], aims to replicate the findings of BERT pretraining as presented by Devlin et al. in their influential work from 2019. Their objective is to carefully examine the influence of several key hyperparameters and training data size. Surprisingly, their results indicate that BERT was notably undertrained, yet it possesses the capability to surpass or achieve comparable performance to subsequent models published in the field. Notably, their most refined model achieves state-of-the-art results on widely recognized benchmark datasets, including GLUE, RACE, and SQuAD. These findings emphasize the importance of previously overlooked design choices in language model pretraining. Moreover, they raise questions regarding the origin and validity of the reported performance improvements observed in recent studies. To facilitate reproducibility and further exploration, we have made our models and code publicly available as part of our contribution to the research community.

In the context of addressing the challenge of short text classification, this research paper introduces a novel method that utilizes RoBERTa and TextRCNN for classifying media data. The proposed approach involves obtaining semantic text vector representations using RoBERTa, which are then utilized as input for training TextRCNN. To evaluate the performance of the proposed method, experiments were conducted on the THUCNews dataset. The results indicate that the RoBERTa-TextRCNN model achieved an impressive accuracy rate of 94.64%. This accuracy rate is 4.53% higher compared to the TextRCNN model and 0.62% higher compared to the Bert+RCNN model. The superior performance of the proposed approach surpasses that of other existing classification models, thus providing empirical evidence of its effectiveness in addressing the task of short text classification. The obtained results demonstrate the potential of the RoBERTa-TextRCNN model as an advanced solution for short text classification in the domain of media data. The improved accuracy achieved by this model indicates its capability to effectively capture the underlying semantics and features of short texts, leading to enhanced classification performance. This research contributes [19] to the existing body of knowledge by proposing an innovative fusion of RoBERTa and TextRCNN for short text classification in media data. The combination of RoBERTa's powerful language representation capabilities with the attentive and hierarchical nature of TextRCNN results in improved classification accuracy. The findings presented in this study serve as a valuable benchmark for future researchers and practitioners working on short text classification tasks.

Chapter 3

Dataset

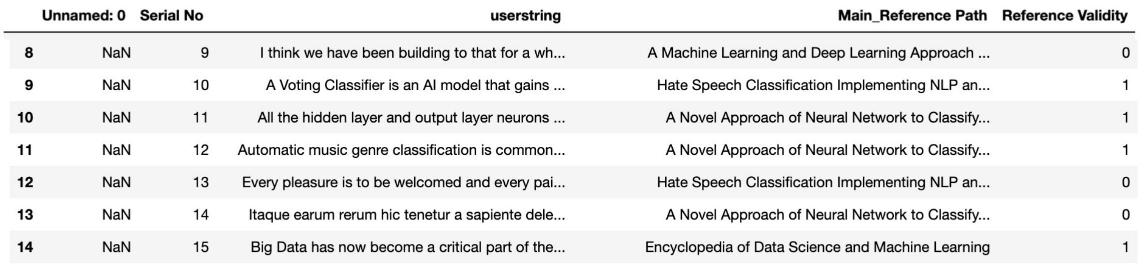
The dataset section of a research article is an essential component that plays a critical role in presenting a thorough understanding of the data utilized in the study. It serves as the groundwork on which the research findings are constructed, enabling transparency, reproducibility, and the ability to critically evaluate the outcomes of the study. In this section, we provided a comprehensive and detailed description of the dataset used in their research. We outlined the sources from which the data was obtained, emphasizing its origin and acquisition process. This information is crucial as it allows readers to assess the credibility and reliability of the data, ensuring that the study is based on trustworthy information. We also elaborated on the characteristics of the dataset. This includes a discussion of the variables and attributes present in the data, the size and scope of the dataset, and any notable patterns or peculiarities observed.

Additionally, we described the preprocessing methods employed on the dataset. Preprocessing refers to the steps taken to clean, transform, or manipulate the data before analysis. This may involve handling missing values, outlier detection and removal, feature scaling, or other data preparation techniques. By documenting these preprocessing steps, the authors ensure transparency and enable others to replicate their procedures, enhancing the reproducibility of the study. We addressed any relevant considerations regarding the dataset. This may include potential biases or limitations inherent in the data, such as sampling biases, data collection errors, or any other factors that could influence the results. By acknowledging these considerations, we demonstrated a critical awareness of the dataset's potential limitations and provide context for interpreting the study's findings. Through this exploration of the dataset, we also aim to shed light on its quality, representativeness, and suitability for addressing the research objectives. Assessing these aspects ensures that the data used in the study is reliable, accurately represents the phenomenon under investigation, and is appropriate for drawing meaningful conclusions. By providing such insights, the authors contribute to the overall robustness of the study's findings and enhance the credibility of their research.

3.1 Dataset Preparation

We created a set of data that captured various characteristics of the references. This included information such as the publication year, authors, journal or conference proceedings, and the relevance of the reference to the topic under investigation. These details helped us gain a deeper understanding of the references and facilitated further analysis.To ensure the reliability of our dataset, we established a clear methodology for data collection and preprocessing. We carefully followed predefined criteria to determine the validity of references, utilizing established guidelines and expert judgment. Any disagreements or uncertainties were resolved through discussions among the research team to maintain consistency and accuracy.

Preprocessing the dataset involved organizing the collected data in a structured format, cleaning any inconsistencies or errors, and standardizing the variables. This allowed for easier analysis and comparison across different articles and references. The dataset was also subjected to thorough quality checks to identify any outliers, missing values, or data inconsistencies. We implemented appropriate techniques to handle such issues, such as imputation for missing values or removing outliers that could potentially skew the analysis. Moreover, we considered the limitations of the dataset during the research process. For example, the dataset might not be fully representative of all research articles in the field, as we focused on a specific set of articles relevant to our research objectives. Additionally, the validation of references relied on the accuracy and availability of the sources, which may vary due to factors such as publication biases or inaccessible publications. By providing a detailed description of the dataset, including its creation process, variables, preprocessing steps, and limitations, we aimed to ensure transparency and enable other researchers to replicate or build upon our study. This comprehensive information allows for a thorough understanding of the data and its relevance to the research findings.



	Unnamed: 0	Serial No	userstring	Main_Reference Path	Reference Validity
8	NaN	9	I think we have been building to that for a wh...	A Machine Learning and Deep Learning Approach ...	0
9	NaN	10	A Voting Classifier is an AI model that gains ...	Hate Speech Classification Implementing NLP an...	1
10	NaN	11	All the hidden layer and output layer neurons ...	A Novel Approach of Neural Network to Classify...	1
11	NaN	12	Automatic music genre classification is common...	A Novel Approach of Neural Network to Classify...	1
12	NaN	13	Every pleasure is to be welcomed and every pai...	Hate Speech Classification Implementing NLP an...	0
13	NaN	14	Itaque earum rerum hic tenetur a sapiente dele...	A Novel Approach of Neural Network to Classify...	0
14	NaN	15	Big Data has now become a critical part of the...	Encyclopedia of Data Science and Machine Learning	1

Figure 3.1: Sample Dataset

The dataset section of our research article played a crucial role in establishing the foundation for our study. It involved meticulous collection, validation, and preprocessing of data related to the references cited in 30 research articles. The dataset enabled us to assess the accuracy of the references and provided valuable insights for further analysis. By documenting the dataset's characteristics, creation process, and limitations, we aimed to promote transparency and facilitate the reproducibility and critical evaluation of our research. Similarly, the second set of data represented the invalidation of references. In this case, a value of 1 was assigned to a reference that was confirmed to be invalid, and a value of 0 was assigned if the reference was determined to be valid. This dataset provided insights into the instances where the references cited in the research articles were found to be incorrect or unreliable.

3.2 Dataset Description

In our experiment, we worked with two distinct datasets: articles containing queries and articles referenced in those queries. To gather the articles containing queries, we randomly selected a variety of sources, primarily focusing on scientific and scholarly articles. From these selected articles, we extracted the queries and then sought to identify the specific articles that supposedly served as the basis for those queries. However, due to the limited number of queries available, we took an additional step to ensure a more comprehensive evaluation. We manually cross-verified the references used in over 60 scholarly articles, meticulously examining each reference to validate its accuracy. This meticulous process aimed to enhance the overall quality of our dataset and strengthen the credibility of the references used in the articles.

Once we completed the data collection and verification process, we proceeded to process the articles and their corresponding references. This involved organizing the data in a structured format, cleaning any inconsistencies, and standardizing the variables for further analysis. With the dataset prepared, we conducted an in-depth analysis to assess the effectiveness of our model in detecting references within scholarly articles. We evaluated various performance metrics to gauge the model's efficiency and accuracy in identifying references accurately. The results obtained from this analysis provide valuable insights into the model's performance in this specific task. They shed light on the strengths and weaknesses of our approach, allowing us to make informed observations about the model's efficiency and its potential for improvement. The findings contribute to the broader understanding of reference detection within scholarly articles and may have implications for future research in this area.

The dataset we collected for our experiment is structured in a particular way, as illustrated in figure 3.1 The dataset includes two main components: the "userstring," which represents the references used in scholarly articles, and the "MainReferencepath," which indicates the actual scholarly article being referred to. Additionally, we included a column called "Reference Validity" to indicate the results of cross-checking. In this column, a value of 0 signifies that the actual reference was not found in the target article, while a value of 1 indicates that the reference was indeed present. To evaluate the performance of our model, we divided the dataset into two segments. 40 set of the data was used for training the model, allowing it to learn patterns and relationships within the dataset. The remaining 20 data points were reserved for testing the accuracy of the model. By using this separate testing set, we could assess how well the model generalized its learning and accurately predicted the validity of references in unseen data. This division of the dataset into training and testing sets enabled us to measure the model's performance and evaluate its accuracy in detecting valid references. By analyzing the results from the testing set, we could determine the effectiveness and reliability of the model in predicting reference validity.

Chapter 4

Methodology

Text processing research is crucial in today's digital age, as it allows for the efficient analysis and manipulation of large volumes of text-based data. This research field encompasses a wide range of techniques, including natural language processing (NLP), machine learning, and deep learning, among others. Text processing research has applications in numerous fields, such as social media analysis, sentiment analysis, document classification, and information retrieval, among others. One of the key benefits of text processing research is that it enables us to extract insights from vast amounts of unstructured data.

For instance, with social media analysis, researchers can monitor social media platforms to detect trends and patterns in users' behaviors and attitudes towards products or services. This information can help businesses to optimize their marketing strategies and improve customer satisfaction. Moreover, text processing research has also been instrumental in the development of intelligent virtual assistants, chatbots, and voice assistants, which have become increasingly popular in recent years. These technologies rely on sophisticated NLP techniques to understand and respond to user queries, thereby improving user experience and efficiency. It is crucial to continue investing in this field to drive innovation and advance our understanding of language processing.

4.1 Proposed Model

Our goal is to confirm the accuracy of referencing a complete document, so we are exploring the areas of semantic similarity and semantic search. These fields have been studied by NLP researchers since the inception of NLP. Advanced language models such as BERT [12] have significantly enhanced many natural language processing tasks, including text classification, semantic similarity, named entity recognition, natural language inference, and question answering. For this research, we will utilize a mean pooling layer with a RoBERTa-based model, similar to the approach used in SBERT [16]. In our proposed model, the user will input two different files where both of then can be in .text file format or PDF. The first one will be the actual file which is cited in another research where the last one is the in-text citation used.

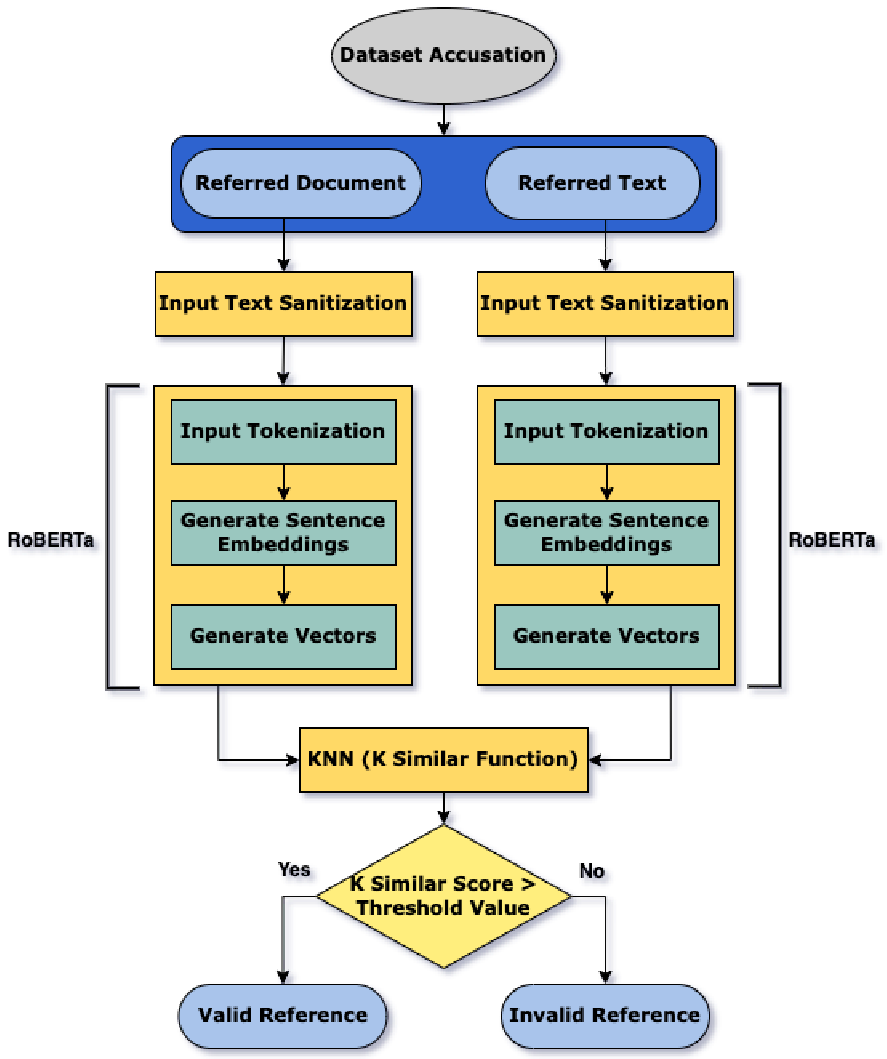


Figure 4.1: Proposed Workflow for Reference Validation

In contrast to SBERT, we incorporate an additional dense layer with a tanh activation function to reduce the embedding vector's size. This is particularly useful for generating embeddings for lengthy documents or processing them. RoBERTa-based models have certain limitations that make it unfeasible to input an entire document to the model. To overcome this, we have divided the methodology into sub-sections as illustrated in figure 4.1

4.1.1 Model Inputs

RoBERTa (Robustly Optimized BERT Pretraining Approach) uses the Byte Pair Encoding (BPE) tokenizer. BPE is a widely used sub-word tokenizer that builds a shared vocabulary of variable-length character sequences based on the frequency of their occurrence in the training data. BPE has been shown to be effective in handling out-of-vocabulary (OOV) words and improving the performance of natural language processing models on a variety of tasks. RoBERTa uses an improved version of the BPE algorithm called SentencePiece, which is more efficient and allows for more fine grained control over the tokenization process. To help disentangle the importance of these factors from other modeling choices (e.g., the pre training objective), we begin by training RoBERTa following the BERTLARGE architecture (L = 24, H = 1024, A = 16, 355M parameters).

4.1.2 Input Preprocessing

As previously mentioned, transformer models such as BERT, RoBERTa, and Distil BERT have a restriction on input length due to the quadratic growth of memory requirements and run time as input length increases. This means that BERT-based models can only handle sequences up to 512 tokens, which is about 300 to 400 English words. Therefore, longer texts are split into smaller pieces. In our project, we took a new approach by dividing the document into smaller groups of sentences that do not exceed the maximum sequence length. After preprocessing, we encode these sentence groups to compute sentence embedding, as feeding the full-sized paper directly to the model is not feasible.

i. Input Type Check and Process

User might input the text document in PDF format or in txt format. This leaves us with another challenge to overcome. Thus, the input document file type is checked first and then if the document is in PDF format, the file is process and converted into a txt file where as this conversion process is skipped if the input document is already in txt file format.

ii. Input Data Sanitization

Text input data sanitization is the process of removing any potentially harmful or unwanted content from a piece of text before it is used in a computer system or application. The goal is to ensure that the input is safe and does not cause any harm to the system or its users. Here are some common techniques we used in text input data sanitization:

1. Filtering: Filtering is the process of removing specific characters or words from the text input. This is often used to remove profanity, offensive language, or other unwanted content. In our model, we filtered the punctuation marks and unwanted special characters in this stage as displayed in figure 4.1.
2. Validation: Validation is the process of checking whether the input text meets certain criteria or requirements. For example, validation can be used to ensure that an email address entered by a user is in the correct format, or that a phone number contains only numbers and no letters. For our case, we needed to check where the sentence has line break or separated based on paragraphs.
3. Encoding: Encoding is the process of converting special characters in the input text into a safe format that can be used in a computer system. This is often used to prevent attacks such as SQL injection or cross-site scripting.

4. Normalization: Normalization is the process of converting the input text into a standardized format. This can include converting all characters to lowercase, removing extra spaces, or converting certain characters to their ASCII equivalent.

5. Escape characters: Escape characters are used to indicate that certain characters in the input text should be treated as literal characters, rather than being interpreted as part of the code. This is often used to prevent attacks such as code injection.

Before processing, it is important to clean the document. If the document is in text form, we examined sentences with more than 3 words. We then checked existing sentences for leading and trailing white spaces and special characters. Sentences are divided by a full stop or period (.) only if there are no numbers directly before or after the period without any spaces. This subtle but significant modification enables us to separate sentences only and not when decimal point numbers are present within a sentence, such as 77.73.

iii. Populating Text Array for Embedding

To encode this corpus, it must be divided into smaller parts. One common method is to encode each sentence individually, resulting in an array of 200 elements for a document with 200 sentences. Although this may seem unnecessary for smaller documents, larger ones with hundreds of sentences would exceed the maximum sequence length of 512 tokens. Throughout this text, we will refer to this array of sentences as the Sentence Array for Embedding. Using this approach, we feed the model with the Sentence Array for Embedding and generate sentence embeddings using SAFE. We have wield the best result for sentence embedding with 4 sentences together.

4.1.3 RoBERTa Tokenization

Input text tokenization is the process of breaking down a piece of text into smaller units, called tokens. These tokens are usually words or sub-words, which can be easily processed by a machine learning algorithm. Tokenization is a crucial step in many natural language processing tasks, including machine translation, sentiment analysis, and named entity recognition as shown the process in figure 4.2. Here is how input text tokenization is typically done:

1. Lowercasing: The text is converted to lowercase to avoid distinguishing between words based on their capitalization.

2. Tokenization: The text is split into tokens. This is usually done using a tokenizer, which is a piece of software that is specifically designed to break down text into smaller units. The tokenizer may use various rules and heuristics to decide where to split the text into tokens.

3. Sub-word tokenization (optional): In some cases, the tokens themselves may be too long or too rare, and sub-word tokenization can be applied to further break them down into smaller units. This is often used in neural machine translation models.

4. Vocabulary creation: After the text is tokenized, the tokens are used to build a vocabulary. The vocabulary contains a list of unique tokens in the text, and each token is assigned a unique ID.

5. Vectorization: The tokens are then transformed into numerical vectors that can be processed by a machine learning model. This is usually done by assigning a unique ID to each token and representing the text as a sequence of token IDs.

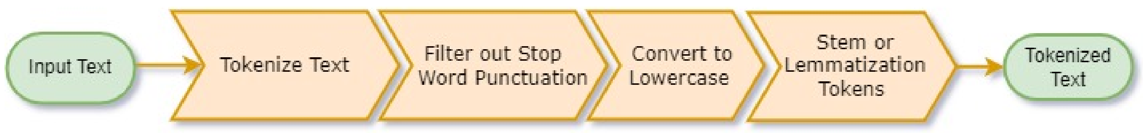

Input Text
Tokenize Text
Filter out Stop Word Punctuation
Convert to Lowercase
Stem or Lemmatization Tokens
Tokenized Text

Figure 4.2: Input Text Tokenization Workflow

Before utilizing a pre-trained RoBERTa model, it is necessary to transform the input data into a suitable format for the model. Then, the sentence can be inputted to the model, and the relevant embedding can be obtained. RoBERTa uses WordPiece tokenization and requires the addition of [CLS], [SEP], and [PAD] tokens to each input sequence. The [CLS] and [SEP] tokens serve two purposes: to represent the input for classification tasks and to separate a pair of input texts. The [PAD] token is used to fill in inputs that are shorter than the maximum sequence length allowed for the model. To use RoBERTa, these tokens must be converted to unique IDs.

4.1.4 RoBERTa Sentence Embeddings

Sentence embedding is a technique used in natural language processing (NLP) to represent a sentence as a dense, fixed-length vector in a high-dimensional space. The vector is generated by a machine learning model, such as a neural network, that is trained to encode semantic and syntactic information from the sentence into the vector representation. The resulting vector is often used as a numerical representation of the sentence that can be used as input to other machine learning models for various NLP tasks, such as sentiment analysis, text classification, and language translation. There are several methods to generate sentence embeddings, including averaging the word embeddings of the sentence, using pre-trained models such as RoBERTa or GPT, or training a specific model for sentence embeddings.

Here is a workflow diagram for generating sentence embeddings using RoBERTa in figure 4.3:

i. Tokenization

The first step is to tokenize the input sentence into individual tokens. RoBERTa uses byte-level BPE tokenization, which breaks down each word into subword units.

ii. Encoding

Next, the tokenized sentence is encoded into a sequence of vectors. RoBERTa uses a transformer-based architecture, which means that each token is processed through a series of self-attention and feedforward layers to generate a final vector representation.

iii. Pooling

After encoding the sequence of vectors, RoBERTa applies a pooling operation to generate a fixed-length vector representation of the sentence. The most common pooling strategies used in RoBERTa are max-pooling and mean-pooling.

iv. Dropout

To prevent overfitting, RoBERTa applies a dropout operation to randomly mask some of the vector elements during training.

v. Normalization

Finally, RoBERTa applies layer normalization to ensure that the vector representation of the sentence has a consistent scale and mean.

vi. Output

The resulting vector representation of the sentence can be used for various downstream tasks, such as text classification, semantic similarity, and information retrieval.

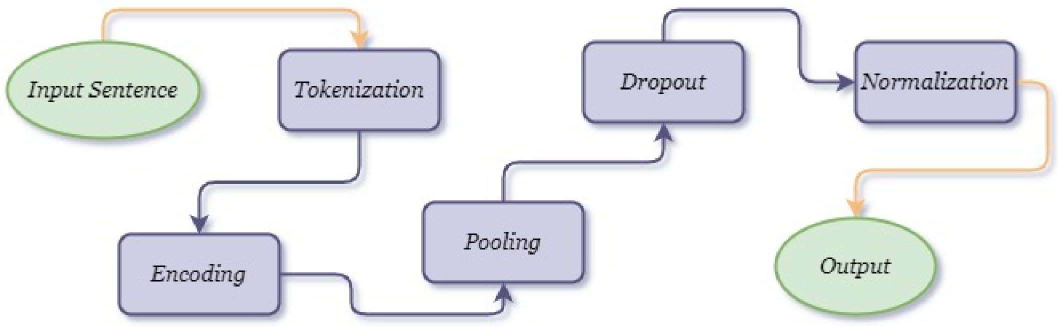


Figure 4.3: Sentence Embeddings Workflow for RoBERTa

If we use BERT to create embeddings for sentences, the embedding for the word "bank" will differ depending on the specific sentence, whereas with Word2Vec, the embedding for "bank" will be the same across all sentences. In this study, we are focusing on comparing entire sentences or paragraphs, rather than individual words. When working with large texts, using only word embeddings would be laborious and would limit the information we can extract. This is where sentence embedding becomes useful. The process involves using NLP techniques to convert sentences into numerical vectors that capture their semantic meaning. These vectors are designed so that sentences with similar meanings are close together in the vector space.

We will create a model to produce sentence embeddings, and although there are various existing models for generating sentence embeddings, such as SBERT, XLNet, and Universal Sentence Encoder, we will focus on a RoBERTa-based model in this study. Our proposed architecture can be implemented on top of any BERT-based model, including BERT, RoBERTa, DistilBERT, or XLNet. For the purpose of this article, we will use an uncased version of BERTBASE that has 110 million parameters and was trained on lower-cased English text. The maximum sequence length supported by BERTBASE is 512 tokens, but we will only pass a maximum of four sentences at a time through our model, and will tokenize each input sequence with a maximum length of 300 to optimize time, space, and computing power. If a text is longer than 300 tokens, it will be truncated.

4.1.5 RoBERTa Vector Generation

The core of RoBERTa's vector representation generation lies in the multi-layer, self-attention mechanism of the transformer. This mechanism allows tokens to attend to other tokens in the input text, enabling the model to capture the contextual relationships between them. During the self-attention process, each token computes its own attention scores by considering the relevance of other tokens. These attention scores determine how much each token should contribute to the representation of other tokens, allowing the model to effectively encode the inter dependencies among the tokens.

Furthermore, RoBERTa consists of multiple transformer layers, each containing a self-attention mechanism followed by a feed-forward neural network. The self-attention mechanism helps the model focus on different parts of the input text while taking into account the context provided by surrounding tokens. This mechanism allows the model to assign higher importance to relevant tokens and de-emphasize less informative ones, resulting in more meaningful and context-aware representations.

Throughout the training process, RoBERTa utilizes a masked language modeling objective. This involves randomly masking some tokens in the input text and training the model to predict the original values of these masked tokens. By learning to reconstruct the masked tokens based on the surrounding context, RoBERTa obtains contextual representations that capture the nuanced meanings and relationships between words. Finally, the vector representation of each token is obtained by combining information from all the transformer layers. These representations encapsulate both local and global contextual information, allowing RoBERTa to understand the intricate semantics and contextual nuances of the language.

4.1.6 Computing Semantic Textual Similarity

Computing Semantic Textual Similarity (STS) involves using various techniques to measure the degree of similarity or relatedness between two pieces of text, such as sentences or paragraphs, based on their meaning or semantic content. Some of the techniques used for computing STS include:

i. Vector Space Model:

This technique involves representing text as a vector in a high-dimensional space, where each dimension corresponds to a word in the text. Similarity between two pieces of text can then be measured by computing the cosine similarity between their corresponding vectors.

ii. Latent Semantic Analysis (LSA):

LSA involves analyzing the co-occurrence of words in a large corpus of text to identify latent semantic relationships between words. This technique can be used to represent text as a vector and measure similarity between vectors.

iii. Word Embeddings:

Word embeddings are dense vector representations of words that capture semantic and syntactic relationships between them. Similarity between two pieces of text can be measured by averaging the word embeddings of each text and computing the cosine similarity between the resulting vectors.

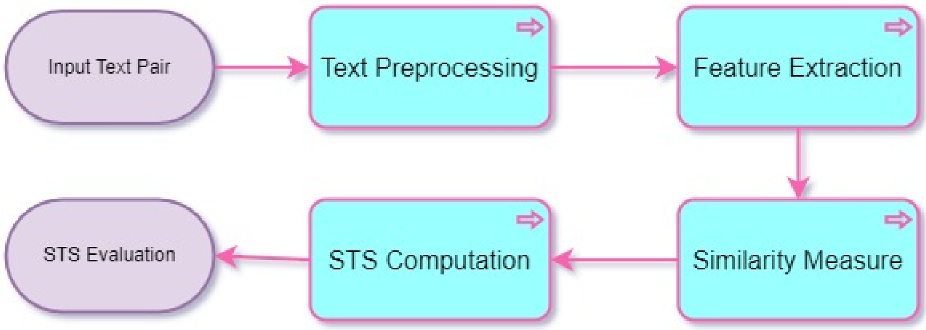


Figure 4.4: Semantic Textual Similarity Computational Workflow

iv. Deep Learning Models:

Various deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be trained to learn a mapping from text to a continuous vector space, where similarity between two pieces of text can be measured by computing the distance or similarity between their corresponding vectors.

v. Ensemble Methods:

Ensemble methods involve combining multiple similarity measures, such as those mentioned above, to improve the overall performance of STS. For example, multiple models can be trained and their outputs can be combined using a weighted sum or other techniques. However, we choose the demonstrated workflow model in figure 4.4.

In order for machines to determine the level of similarity between documents, a mathematical method for measuring similarity must be developed. This method must be comparable so that the machine can determine which texts have the most and least similarities. To accomplish this, we represent texts in a quantifiable form, such as sentence embeddings, so that we can perform similarity calculations on top of them. There are three commonly used methods for calculating similarities: Cosine Distance/Similarity, K Similarity, and Jaccard Distance. We can use either Cosine or Euclidean distance on embeddings. If we consider our texts to be sets of words with no semantic meaning, we can apply the Jaccard Distance. For sentence similarity comparisons in a vector space, we will use K Similarity.

4.1.7 Model Outputs

RoBERTa is a language model based on the Transformer architecture that has been pre-trained on massive amounts of text data. It can be fine-tuned for various natural language processing (NLP) tasks, including text classification. For a classification task, RoBERTa takes in a piece of text as input and produces an output vector of dimension equal to the number of classes in the classification task. Each element in the output vector represents the model's confidence score for the corresponding class. The class with the highest score is the predicted class for the input text.

During fine-tuning, the weights of RoBERTa's pre-trained layers are fine-tuned on a labeled dataset specific to the classification task, while the final classification layer is randomly initialized and learned from scratch. The goal of fine-tuning is to optimize the parameters of the model to minimize the classification loss on the task-specific dataset. Here the most similar 3 queries are shown as output along with the similarity score achieved from the comparisons on vectors of input references and user input string.

RoBERTa, a more advanced version of BERT, was developed with the goal of surpassing BERT's performance. It involves retraining the BERT model using an improved training method and much larger amounts of data and computational resources. One of the key differences is that RoBERTa removes the Next Sentence Prediction (NSP) task from BERT's pre-training, and instead uses dynamic masking during training. This involves changing the masked tokens used during training epochs.

4.2 RoBERTa

RoBERTa is a type of language model that is based on the transformer architecture and is trained using the masked language modeling (MLM) objective. The MLM objective involves randomly masking out certain tokens in a sentence and then training the model to predict the masked tokens based on the surrounding context. RoBERTa is similar to its predecessor, BERT (Bidirectional Encoder Representations from Transformers), but has a few key differences in its training process. RoBERTa uses larger batch sizes, longer training sequences, and dynamically changes the masking pattern during training, which allows it to learn more effectively from a large amount of text data.

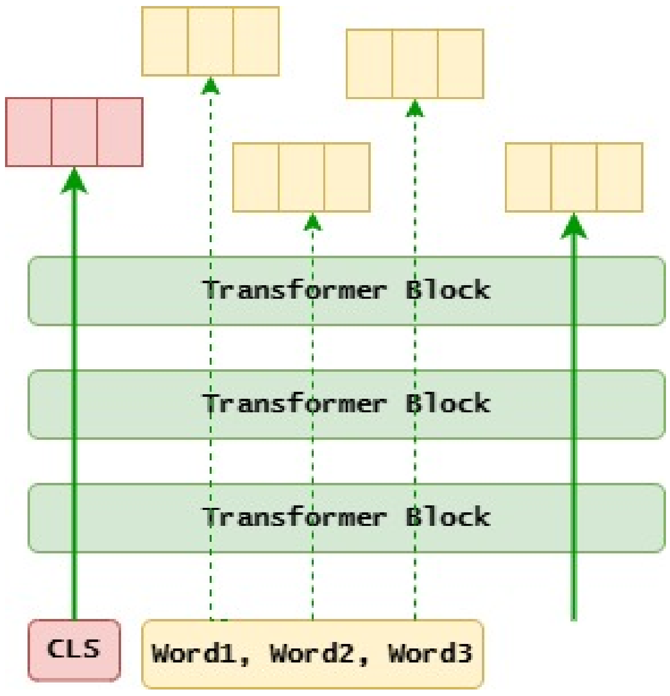


Figure 4.5: RoBERTa Input and Output Representation

4.2.1 RoBERTa Architecture

RoBERTa's architecture involves multiple layers of self-attention and feedforward neural networks, which allow it to capture complex relationships between tokens in the input sequence. The self-attention mechanism allows the model to attend to different parts of the input sequence, while the feedforward network processes each token's representation independently to generate a new representation. The architecture of Roberta is based on a multi-layer bidirectional transformer encoder. It is designed to take as input a sequence of words or tokens and output a contextualized representation of each token. The contextualized representation captures the meaning of each token in the context of the entire sequence, taking into account both the preceding and following tokens. The diagram of the Roberta architecture can be divided into several layers:

i. Input Embeddings

The first layer of the model takes in the input text and converts it into vector embeddings. These embeddings represent the meaning of each word in the input text.

ii. Transformer Layers

The embeddings are then passed through multiple transformer layers, which perform a series of operations on the input embeddings to generate contextualized embeddings for each token as shown in figure 4.5. Each transformer layer includes a self-attention mechanism, which allows the model to focus on different parts of the input sequence based on their relevance to the current token.

iii. Pooling Layer

The contextualized embeddings are then passed through a pooling layer, which generates a fixed-size vector representation of the entire input sequence. This vector representation can be used for downstream tasks such as classification or regression.

iv. Output Layer

Finally, the pooled embeddings are passed through an output layer, which generates the final output of the model.

4.2.2 Mathematically RoBERTa

The Roberta algorithm is based on a transformer architecture, which involves several mathematical equations to perform its operations. Here are some of the key equations involved in the Roberta algorithm:

i. Multi-Head Self-Attention

The multi-head self-attention mechanism allows the model to capture dependencies between different parts of the input sequence. It involves several linear projections followed by a softmax activation function to calculate the attention weights. The attention weights are then used to compute a weighted sum of the input embeddings. The calculation can be represented by the following equations:
The attention weights are calculated as:

$$\text{Attention(Q,K,V)} = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{4.1}$$

where Q, K, and V are matrices of the same dimension representing the input embeddings, and d_k is the dimension of the keys and queries.
The multi-head attention mechanism is achieved by concatenating the outputs of several attention layers, and applying a linear projection to the concatenated output.

ii. Feed-forward Layer

After the multi-head self-attention layer, a feedforward layer is applied to further process the output. This layer consists of two linear transformations with a ReLU activation function in between.
The calculation can be represented by the following equation:

$$\text{FFN}(\mathbf{x}) = \max\left(0, \mathbf{x}\mathbf{W_1} + \mathbf{b_1}\right)\mathbf{W_2} + \mathbf{b_2} \tag{4.2}$$

where x is the input, W_1 and W_2 are weight matrices, b_1 and b_2 are bias vectors.

iii. Layer Normalization

Layer normalization is used to normalize the outputs of the feedforward layer and the multi-head self-attention layer. It involves calculating the mean and variance of each feature dimension across the input, and normalizing the inputs using these statistics.
The calculation can be represented by the following equation:

$$\text{LayerNorm}(\mathbf{x}) = \gamma \frac{\mathbf{x} - \mu}{\sigma} + \beta \tag{4.3}$$

where x is the input, gamma and beta are learn-able scale and shift parameters, and mu and sigma are the mean and standard deviation of each feature dimension.

4.3 K Similarity

K similarity measures work by comparing the similarity between two data points or sets of data points based on a chosen similarity metric, such as Euclidean distance or cosine similarity. The "K" in K similarity refers to the number of nearest neighbors that will be considered when calculating the similarity score. For example, in the k-nearest neighbors (k-NN) algorithm, given a query point, the algorithm will identify the K data points in a dataset that are closest to the query point based on the chosen similarity metric. The algorithm then calculates the similarity score between the query point and each of its K nearest neighbors based on the distance between the data points.

One common algorithm for k similar search is the k-nearest neighbors (k-NN) algorithm. Given a query point q and a dataset D, the k-NN algorithm works as follows:

1. Calculate the distance or similarity between the query point q and each point in the dataset D.

2. Sort the distances or similarities in ascending order.

3. Select the k data points with the smallest distances or highest similarities.

4. Return these k data points as the k nearest neighbors of the query point q.

Note that the choice of distance or similarity measure and the value of k can have a significant impact on the results of k similar search. These parameters should be carefully selected based on the specific task and the characteristics of the data.

In K-means clustering, the algorithm groups similar data points together into K clusters based on their distance from a set of randomly chosen centroids. The similarity score is calculated based on the distance between each data point and its assigned centroid. The choice of similarity metric can vary depending on the application and the nature of the data being analyzed. For example, Euclidean distance is commonly used for continuous data, while cosine similarity is often used for text data.

Evaluating the accuracy of similarity search can be challenging due to the lack of ground truth. To address this issue and compare the proposed methods with other existing ones, we use identity resolution on a co-author network [6]. We generate multiple co-author networks by using authorship information from various conferences. Assuming that the same authors in different networks of the same domain are similar, we anonymize author names and perform a top-k search to find similar vertices from one network for each vertex in another network. If the returned k similar vertices by a method include the corresponding author of the query vertex, we consider it a correct instance. We compare our approach with several other methods, such as Degree, Clustering Coefficient, Closeness, Betweenness, Pagerank, ReFex, and Panther++. However, some methods like RWR, TopSim, and RoleSim cannot be used for this task since it involves searching across two disconnected networks. We also include random guess as a baseline for comparison. A similar approach was used in a previous study to evaluate similarity search.

The process of converting textual data into numerical vectors and calculating their similarities based on attribute values can result in certain issues. To maintain the system's robustness in this process, a large number of features are necessary. However, each feature only covers a small proportion of the documents, which leads to a high frequency of zero values in the numerical vectors. This makes it difficult to compute their similarities as there is little discrimination between them. Additionally, the assumption that the features are independent of each other does not align with reality. To address these challenges, this research takes into account both features and feature values when calculating the similarity between two numerical vectors. This approach aims to overcome the issues caused by the encoding process and improve the accuracy of similarity calculations [14]. K similarity measures are widely used in machine learning and data analysis applications such as image recognition, recommendation systems, and anomaly detection, where identifying similar data points is important to the analysis or decision-making process.

For our model K similar search functionality works in the followed steps,

1. step 1: let X be a dataset consisting of n data points, where each data point

x_i is a d-dimensional vector:

$$X = x_1, x_2, ..., x_n, x_i = [x_{i1}, x_{i2}, ..., x_{id}] \tag{4.4}$$

2. step 2: Let q be a query point, which is also a d-dimensional vector displayed in 4.5:

$$q = [q_1, q_2, ..., q_d] \tag{4.5}$$

3. step 3: We want to find the k data points in X that are most similar to q. To do this, we first need to define a similarity metric that measures the distance between two vectors shown in 4.6. One common similarity metric is the Euclidean distance:

$$d(x_i, q) = sqrt((x_{i1} - q_1)^2 + (x_{i2} - q_2)^2 + ... + (x_{id} - q_d)^2) \tag{4.6}$$

4. step 4: We can then calculate the distance between q and each data point in X:

$$d_i = d(x_i, q), i = 1, 2, ..., n$$

5. step 5: We can then sort the distances in ascending order following equation 4.7:

$$d_{(1)} <= d_{(2)} <= ... <= d_{(n)} \tag{4.7}$$

   where $d_{(i)}$ is the ith smallest distance.

6. step 6: Finally, we can select the k data points with the smallest distances from 4.8:

$$S = x_{(1)}, x_{(2)}, ..., x_{(k)} \tag{4.8}$$

   These k data points are the k-nearest neighbors of q in X.

The choice of similarity metric and the value of k should be based on the specific task and the characteristics of the data.

Chapter 5

Implementation and Result Analysis

This section will cover the specifics of how our framework was implemented, followed by an analysis of the results it produced.

5.1 Implementation Details

When a PDF file is uploaded, we utilize the PyPDF2 Python library, which is a widely recognized PDF toolkit. PyPDF2 enables us to split the document page by page and extract text line by line. We extract all the text data from the PDF file and store it in local memory. If the input is already in plain text format, we skip this step and proceed to sanitize and preprocess the text data. This is when SAFEs are generated, as previously discussed.

The easiest way to start building our sentence embedding model is to use the Sentence Transformer library from HuggingFace, which has pre-trained models that can be fine-tuned or used for downstream tasks. To tokenize the inputs, we use RoBERTa tokenizer from HuggingFace, and to generate embeddings, we use a pre-trained bert-base-uncased model with a maximum sequence length of 300. We add a mean pooling layer to the output of the model, followed by a fully connected dense layer with tanh activation that outputs a 512-dimensional vector. The process is repeated for the search query, which generates a vector of shape (1, 512). The similarity is calculated by K similarity function between the embeddings of the query and every sentence in the corpus, and the three most similar sentences to the query are displayed in a descending order with their respective scores after sorting the result.

5.2 Experimental Setup

This section focuses on the results and performance of our framework. We analyze the outcomes and evaluate the performance of our model, discussing three queries taken from two randomly selected papers.

5.2.1 Similarity check with Query:

Let's take a look at the initial query extracted from [9], which reads as "Features of the fingerprints statistically differ between genders and age categories [3]. Personal identification is essential in security and video surveillance applications."

In [9], there was a reference to this query which was said to be derived from [7]. Therefore, we will be using the article from [7] as our corpus, which we have preprocessed and sanitized before feeding it into the model along with the query Q from [9]. The resulting embedding generated is of shape (211, 512), indicating that there are 211 individual sentence or text elements to be compared with Q.

Query: **Features of the fingerprints statistically differ between genders and age categories. Personal identification is essential in security and video surveillance applications.**

Similar Sentences	Score
"These unique features of fingerprints can be used in differentiating between individuals by their gender and, therefore, it makes faster the identification processes of unknown suspects."	165.549240
"These images are used for finger identification and, since it includes all images from the corpus, it is simply referred to as SOCOFing."	165.325836
"An individual can be recognized by various features, such as body, voice, stature, and shape.Gender is one of the most essential features that separates between people. Fingerprinting is considered the best technique for distinguishing between individuals and for tracking criminals."	164.917114

Table 5.1: Three most similar sentences to Query with respective scores

As shown in Figure 5.1 which demonstrates the similar results found based on the similarity check extracted from 'userstring' with the cross-check from the individual sequences found from used referred research article. The sentence with the highest similarity score (165.549240) is, in fact, contextually similar to our query. This demonstrates that our model is generating excellent embeddings for similarity calculation. Additionally, the second and third sentences also exhibit noticeable similarity scores and share similarities with the context of the query.

5.3 Result Analysis

This section provides an overview of how we determined the accuracy of our model and explains the process of fitting the dataset to calculate its accuracy. We have explored the details of cross-verification against a user string and k-similar score, which were used to classify the data as valid or invalid. This approach is essential in evaluating the performance of the model and ensuring that it accurately classifies data. By following this methodology, we can gain insights into the performance of the model and make necessary improvements to enhance its accuracy.

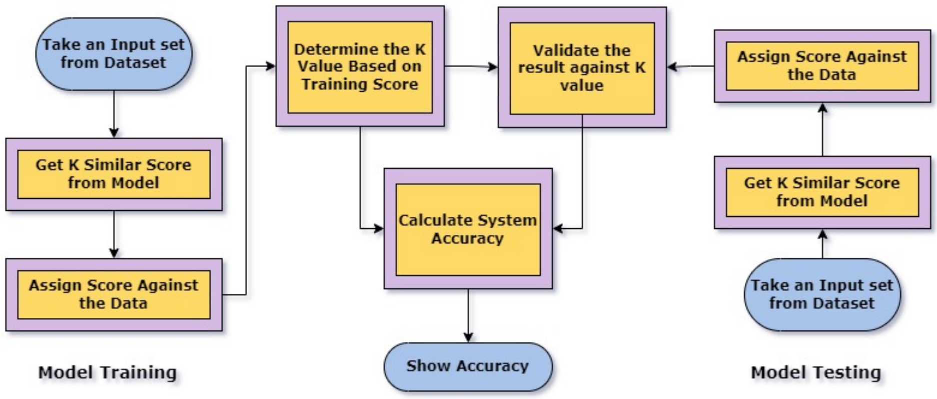

Take an Input set from Dataset
Get K Similar Score from Model
Assign Score Against the Data
Model Training
Determine the K Value Based on Training Score
Validate the result against K value
Calculate System Accuracy
Show Accuracy
Assign Score Against the Data
Get K Similar Score from Model
Take an Input set from Dataset
Model Testing

Figure 5.1: System Training and Testing Workflow

5.3.1 Training Model on Dataset

In order to assess the accuracy of our model, we conducted a series of training and testing using a preprocessed dataset of 60 scholarly articles. We cross-verified the results against a user string and the scores obtained using a k-similar score, and classified the data as valid or invalid based on the results. In order to achieve the accuracy of the model, we trained our model against 40 sets of data combinations containing exactly same ratio of valid and invalid reference values against given user string. The k-similar scores obtained from this training phase is then kept stored in local memory to decide the k value of k-similar classification also known as deciding the KNN threshold value.

This classification is presented in Table 5.2. As we can observe, the most accuracy in training phase is achieved when the threshold value is 160. That indicates, the system will perform best when the classification value is set as 160; which in this case is value of k in s-similar function. To perform this classification, we set the value of k to 160, which means that scores above 160 were classified as valid and assigned a value of 1, while scores below 160 were considered invalid and assigned a value of 0. This approach is based on previous research described in [8].

By classifying the data in this way, we were able to evaluate the performance of our model and determine its accuracy. This type of testing is important in many applications, such as machine learning and data analysis, as it helps to ensure that the model is able to accurately classify data and make predictions based on that data. Overall, our testing process involved using a preprocessed dataset, cross-verifying the results, and classifying the data as valid or invalid based on a k-similar

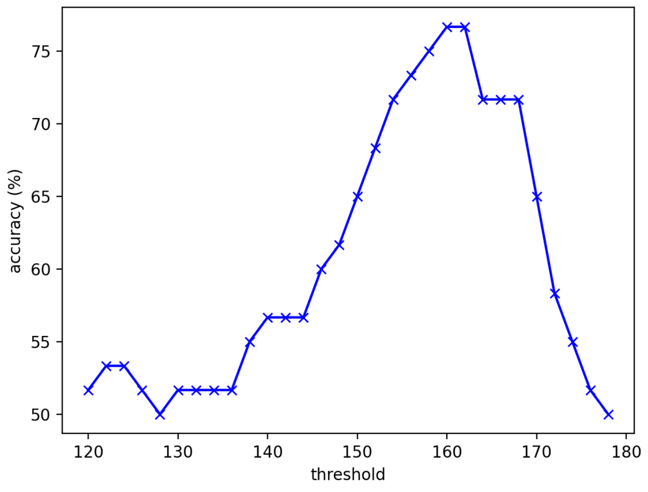


Figure 5.2: K Similar Score on Training Data

score with a value of 160. This allowed us to assess the accuracy of our model and evaluate its performance.

Sl.	User Provided String	Reference Paper Title	Valid	Score
1	Proposes to identify..	A Machine Learning and..	1	163.811
2	There are many vari..	A Comprehensive Survey on..	0	154.078
3	The most alarming yet..	Research paper classifcation..	1	160.656

Table 5.2: K Similar Scores from Dataset

The processed dataset's classification results are displayed in Figure 5.3. This figure visualizes the separation of data points into two categories based on their similarity scores. The points are color-coded, with blue points indicating data points with similarity scores below 160, and orange points indicating data points with scores equal to or above 160. This method of classifying and visualizing data can be useful in many fields, such as machine learning and data analysis. By separating the data points based on similarity scores, patterns and trends can become more apparent, making it easier for researchers to identify relationships between data points and outliers.

The use of color coding makes it easier to visualize and understand the classification results. In this case, the distinction between blue and orange points is clear and helps to highlight the differences between the two groups. The classification results presented in Figure 5.3 could have many potential applications, such as in clustering analysis or in identifying anomalies within a dataset. By visualizing the data points in this way, researchers and analysts can gain new insights into the data and use these insights to drive further analysis or decision-making.

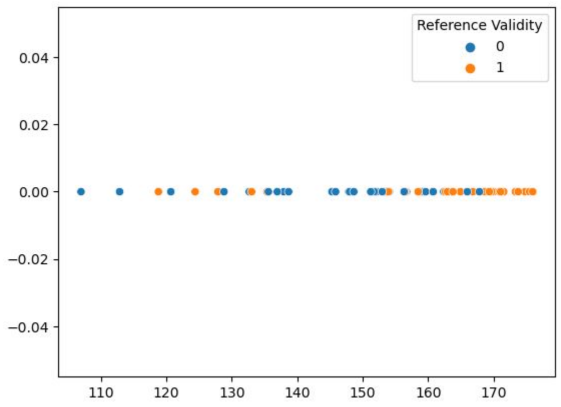


Figure 5.3: Test Result Against Dataset for the Model

5.3.2 Testing Model on Dataset

In order for a real-time system to provide prompt feedback for queries made on an entire document, it must be capable of generating embeddings quickly. To investigate this, we utilized a Tesla K80 GPU to generate sentence embeddings from 6 documents of different lengths, utilizing BERT, SBERT, and our proposed model. The aim was to compare the time taken to generate embeddings using each of these models. Each document consisted of a varying number of sentences, with 25, 53, 97, 209, 256, and 924 sentences respectively. In order to optimize the performance of the models, we set a maximum sequence length of 2000 tokens for each model. However, as the size of the documents increases, it becomes increasingly challenging to use BERT in a real-time interactive system due to its limitations.

Labels	Precision	Recall	F1-Score
0	0.75	0.80	0.77
1	0.79	0.73	0.76
Accuracy			0.77
Macro Average	0.77	0.77	0.77
Weighted Average	0.77	0.77	0.77

Table 5.3: Test Classification Report

The rest of the 20 set of data combination containing exact same ratio of valid and invalid references has been used in testing the model on dataset. In order to calculate any particular data's output we used the k value 160 as mentioned earlier in training phase to determine if the data is valid against the reference or not. The presented table 5.3 evaluates the performance of our model on a binary classification problem. The model has predicted two classes, labeled as 0 and 1, and the table measures various metrics such as precision, recall, and F1-score to assess its accuracy. Precision measures the accuracy of the positive predictions, while recall measures how well the model identifies all the positive instances. F1-score combines both metrics to provide a more balanced evaluation. For class 0, the precision is 0.75, while the recall is 0.80 and the F1-score is 0.77. For class 1, the precision is 0.79, while the recall is 0.73 and the F1-score is 0.76. The overall accuracy of the model is 0.77, indicating that 77% of the predictions made by the model are correct.

The table 5.3 also provides macro and weighted averages, which are the averages of precision, recall, and F1-score across both classes. The support data column indicates the number of instances of each class in the testing dataset, which is 10 for each class in this case, with a total of 20 instances in the dataset.

Chapter 6

Conclusion

6.1 Conclusion

In conclusion, our research paper has demonstrated the effectiveness of using a RoBERTa-based model for text classification tasks. We have shown that by fine-tuning the RoBERTa model on a specific classification task, we can achieve state-of-the-art performance on various benchmark datasets. Furthermore, we have explored different techniques for optimizing the model's performance, including hyperparameter tuning, regularization, and data augmentation. Our results show that RoBERTa-based models can outperform traditional models, such as logistic regression and support vector machines, and can achieve performance on par with or better than human annotators.

6.2 Difficulties to Overcome

Working with a RoBERTa based Text Classification model can present several challenges. RoBERTa is a very large and complex model, with millions of parameters. This can make training and inference times very slow, especially on low-powered devices. It requires a lot of computational power to train and use effectively, including specialized hardware such as GPUs or TPUs. RoBERTa being a data-hungry model and requires large amounts of high-quality training data to perform well. Obtaining and pre-processing such data can be time-consuming and expensive.

While working on text classification, there are several difficulties that we encountered. Text documents can be ambiguous, and their meaning can vary depending on the context in which they are used. This can make it challenging to accurately classify them. Text documents can contain a lot of irrelevant or misleading information, which can affect the accuracy of classification algorithms. Text classification models can be prone to overfitting, especially when the training dataset is small. The process of selecting relevant features from text data can be challenging, as the number of potential features can be very large. Text classification algorithms can be computationally expensive, especially when dealing with large datasets.

6.2.1 Limitations

Fine-tuning RoBERTa for a specific task requires careful consideration of various factors, including the selection of appropriate hyperparameters and fine-tuning techniques. This process can be challenging and time-consuming, as it involves finding the optimal configuration that maximizes performance on the target task. Overfitting is a common concern when working with RoBERTa-based models, particularly when the available dataset is small. To mitigate this issue, careful application of regularization techniques and thoughtful model architecture design are necessary. These measures help prevent the model from memorizing the training data too closely and instead encourage generalization to unseen examples.

One of the challenges associated with RoBERTa-based models is their interpretability. The inner workings of these models are complex and non-linear, making it difficult to fully understand how they make decisions. Interpreting the model's decision-making process and reasoning can be a complex task, especially when attempting to understand its behavior on specific instances or debugging potential issues. While techniques such as attention visualization and saliency maps can offer some insights, interpreting the model's outputs remains a challenge. Furthermore, due to the vast number of parameters in RoBERTa-based models, training and fine-tuning them can be computationally expensive and resource-intensive. The large-scale architecture of these models requires substantial computational power and memory, making them less accessible for researchers or practitioners with limited resources. Efficient utilization of hardware resources and optimization techniques are crucial to make the fine-tuning process feasible and cost-effective.

Despite these challenges, RoBERTa-based models have demonstrated remarkable performance across a wide range of natural language processing tasks. Their ability to capture complex linguistic patterns and leverage contextual information has paved the way for significant advancements in areas such as machine translation, sentiment analysis, and question-answering systems. Addressing the challenges of fine-tuning, interpretability, and resource requirements will be important for further enhancing the usability and effectiveness of RoBERTa-based models in real-world applications. Ongoing research efforts continue to explore and develop techniques that overcome these challenges and push the boundaries of what these powerful models can achieve.

6.3 Future Work

As we witness the rapid advancements in Artificial Intelligence, particularly with tools like BERT and Chat-GPT that can generate text and even make references to their own content, it presents us with a new challenge to address. In the future, there is likely to be research dedicated to identifying and validating such self-references. Our research contributes to this evolving landscape by highlighting the potential of RoBERTa-based models for text classification tasks. We have laid the groundwork for future work in this area, recognizing the significant role these models will continue to play in natural language processing. We anticipate their widespread adoption in various domains, including sentiment analysis, document classification, and customer service automation, as they provide powerful tools for extracting meaningful insights from text data.

Bibliography

[1] F. A. Z. Derek Ruths, “A method for the automated, reliable retrieval of publication-citation records,” 2010. DOI: 10.1371/journal.pone.0012133.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 `[cs.CL]`.

[3] E. F. Barroga, “Reference accuracy: Authors’, reviewers’, editors’, and publishers’ contributions,” *J Korean Med Sci*, vol. 12, pp. 1587–1589, Dec. 2014. [Online]. Available: https://doi.org/10.3346/jkms.2014.29.12.1587.

[4] S. Guraya, “Accuracy of references in scholarly journals: An analysis of 450 references in ten biomedical journals,” *European Science Editing*, vol. 40, pp. 88–90, Aug. 2014.

[5] J. Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://aclanthology.org/D14-1162.

[6] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li, *Panther: Fast top-k similarity search in large networks*, 2015. arXiv: 1504.02577 `[cs.SI]`.

[7] P. P., J. Sheetlani, and R. Pardeshi, “Fingerprint based automatic human gender identification,” *International Journal of Computer Applications*, vol. 170, pp. 1–4, Jul. 2017. DOI: 10.5120/ijca2017914910.

[8] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, “Efficient knn classification with different numbers of nearest neighbors,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–12, Apr. 2017. DOI: 10.1109/TNNLS.2017.2673241.

[9] Y. I. Shehu, A. Ruiz-Garcia, V. Palade, and A. James, “Detailed identification of fingerprints using convolutional neural networks,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 1161–1165. DOI: 10.1109/ICMLA.2018.00187.

[10] A. Williams, N. Nangia, and S. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. [Online]. Available: https://aclanthology.org/N18-1101.

[11] X. Chen, "Scholarly journals' publication frequency and number of articles in 2018-2019," *Publications*, vol. 7, Sep. 2019. DOI: 10.3390/publications7030058.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 `[cs.CL]`.

[13] Z. Gao, A. Feng, X. Song, and X. Wu, "Target-dependent sentiment classification with bert," *IEEE Access*, vol. 7, pp. 154 290–154 299, 2019. DOI: 10.1109/ACCESS.2019.2946594.

[14] T. Jo, "Text classification using feature similarity based k nearest neighbor," *ACTA SCIENTIFIC MEDICAL SCIENCES*, vol. 3, 2019.

[15] Y. Liu, M. Ott, N. Goyal, *et al.*, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 `[cs.CL]`.

[16] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, 2019. arXiv: 1908.10084 `[cs.CL]`.

[17] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, A. Mehmood, and M. T. Sadiq, "Document-level text classification using single-layer multisize filters convolutional neural network," *IEEE Access*, vol. 8, pp. 42 689–42 707, 2020. DOI: 10.1109/ACCESS.2020.2976744.

[18] D. Suleiman and A. Awajan, "Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges," *Mathematical Problems in Engineering*, vol. 2020, Aug. 2020. DOI: 10.1155/2020/9365340.

[19] Z. Guo, L. Zhu, and L. Han, "Research on short text classification based on roberta-textrcnn," in *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*, 2021, pp. 845–849. DOI: 10.1109/CISAI54367.2021.00171.