SAAD DAHLEB BLIDA 1 UNIVERSITY
Faculty of Sciences
Computer Science Departement

Master's Dissertation In Computer Science

Option: *Software Engineering*

Authorship Attribution On Social Networks

Realized by :
BELABBES NABI Mina
SAHI Serine

Supervised by :
Dr. BOUMAHDI Fatima
Mr. REMMIDE Abdelkarim

Members Of The Jury:

President : Dr. OUKID Lamia

Examinator : Dr. REZZOUG Nachida

Academic year : 2022 – 2023

Acknowledgements

We would like to begin by expressing our gratitude to god almighty for guiding us and providing us with the strength and perseverance to complete this Master's dissertation, we are certainly grateful for the blessings bestowed on us throughout this journey.

We extend our heartfelt appreciation to our promotor Mrs.Boumahdi for her unwavering support, mentorship, and expertise. We are immensely grateful for her willingness to accept the responsibility of promoting and supervising our work. Her supervision, guidance, and share of knowledge have been instrumental in successfully completing our dissertation.

We would like to express appreciation to our co-promoter Mr.Remmide for his valuable support and contribution to our work, we are truly grateful for his active involvement and the insights he has provided throughout the process

Furthermore, we are deeply thankful to our parents and families for their unwavering love, support and encouragement. Their belief in us and constant support have been an immense source of strength throughout the entire course of our work.

Finally, we would like to thank the members of our jury for accepting the responsibility of evaluating and judging our work.

Abstract

As social media platforms continue to grow in popularity and influence, the need to address problems related to content integrity and user accountability becomes more critical. Authorship attribution serves as a powerful tool in tackling such issues by accurately determining the real author of online posts.

In this dissertation we propose an approach that utilizes deep learning models including Temporal Convolutional Networks (TCN) and Long Short-term Memory Networks (LSTM) combined with Convolutional neural networks (CNN) , along with machine learning models such as Autoencoder and Adaboost to effectively predict the authors of unknown online posts. To evaluate the effectiveness of our approach, we conducted experiments on various scenarios using a Twitter dataset, where we achieved an accuracy rate of 52.77% in authorship attribution.

Keywords: NLP, Deep Learning, Authorship Attribution.

Résumé

Alors que les plateformes de réseaux sociaux ne cessent de gagner en popularité et en influence, la nécessité de résoudre les problèmes liés à l'intégrité du contenu et à la responsabilité des utilisateurs devient de plus en plus critique. L'attribution d'auteur est un outil puissant pour résoudre ces problèmes en déterminant avec précision l'auteur réel des messages en ligne.

Dans ce mémoire, nous proposons une approche qui utilise des modèles d'apprentissage profond, notamment les réseaux convolutifs temporels (TCN) et les réseaux de mémoire à long terme (LSTM) combinés avec des réseaux neuronaux convolutifs (CNN), ainsi que des modèles d'apprentissage automatique tels que Autoencoder et Adaboost pour prédire efficacement les auteurs de messages en ligne inconnus. Pour évaluer l'efficacité de notre approche, nous avons mené des expériences sur divers scénarios à l'aide d'un ensemble de données Twitter, où nous avons atteint un taux de précision de 52,77% dans l'attribution d'auteur.

Mots Clée : TAL, Aprentissage Profond, Attribution d'auteur.

ملخص

مع استمرار تزايد شعبية منصات التواصل الاجتماعي وتأثيرها ، فإن الحاجة إلى معالجة المشاكل المتعلقة بسلامة المحتوى ومساءلة المستخدمين تصبح أكثر أهمية يُعد إسناد التأليف أداة قوية في معالجة مثل هذه المشكلات من خلال التحديد الدقيق للمؤلف الحقيقي للمشاركات عبر الإنترنت

في هذه الذاكرة، نقترح نهج يستخدم نماذج التعلم العميق، مثل شبكات الذاكرة الزمنية المؤقتة و شبكات الذاكرة القصيرة الأجل مع شبكات العصب الصناعي المتلافية، بالإضافة إلى نماذج التعلم الآلي مثل المشفر التلقائي لتنبؤ بمؤلفي المنشورات الغير معروفة على الإنترنت بفعالية. لتقييم فعالية نهجنا ، أجرينا تجارب على سيناريوهات مختلفة باستخدام مجموعة بيانات تويتر، حيث حققنا معدل دقة بنسبة 52.77% في إسناد التأليف

الكلمات المفتاحية: المعالجة اللغوية الطبيعية، التعلم العميق، إسناد التأليف

Contents

List of Figures

List of Tables

List of Algorithms

General Introduction

Social networks have seen an extraordinary increase over the years, with the number of users on social network platforms reaching 4.8 billion in 2023 [Petrosyan, 2023], having almost tripled in the past decade [Dean, 2023]. It is now possible for individuals to express themselves freely, to share their opinions, their ideas and to interact with other people worldwide.

However, with this rapid rise in social networks usage, comes a great deal of concerns that must not be overlooked. The spread of online harassment has become a major problem, affecting the well being of individuals of all ages, identity theft in social networks is a growing problem that is affecting people more and more everyday. Additionally, intellectual property violations and plagiarism pose a real threat to original authors' credibility with the ease of copying and pasting from online sources.

Problem

Studies have shown that social network platforms are the least trusted source of news and information worldwide [Watson, 2023]. The extensive spread of misinformation, with the ease of sharing unverified content have contributed to the decline of trust on social networks. When confronted with these problems and particularly the growing number of social media users, which poses an additional challenge in accurately identifying the true author and distinguishing their unique patterns, it is crucial to develop robust solutions that can accurately determine the true origin of online content. By addressing these problems, we can establish accountability and create a safe digital environment while promoting responsibility and transparency. This raises important questions: How can we effectively identify the origin of online content? How can we address the growing challenges on social network platforms and how can we navigate the growing complexity introduced by the large number of users online in order to accurately attribute authorship?

Objective

In light of these challenges and questions our work aims to propose three distinct architectures for authorship attribution. We first utilise Temporal Convolutional Networks (TCN) which has not been used before in this task. In addition we investigate the effectiveness of Long-short term Memory Networks (LSTM) combined with Convolutional Neural Networks (CNN). Finally we explore the use of an autoencoder combined with adaboost. In conjunction with these techniques, we utilize a word embedding method namely GloVe to enhance the semantic understanding of the data and capture contextual information important for authorship attribution. We train and evaluate our proposed models using a large twitter dataset, which will further enable them to effectively extract meaningful features and capture unique writing styles present in the data in order to produce accurate predictions for the authorship attribution of unknown tweets.

Dissertation Plan

Our dissertation is organised in three chapters followed by a general conclusion.

Chapter 1: In this chapter we provide an overview of the authorship analysis field and its sub-fields while highlighting the authorship attribution domain which is the focus of our study, we also present the previous works conducted in authorship attribution while outlining their different approaches and methods, we conclude it by providing a comparative analysis of the results obtained.

Chapter 2: In the second chapter we will focus on our proposed approach, including the dataset that has been used and the various pre-processing techniques utilized. Additionally, we provide descriptions of the different models that have been used in our task and their architectures.

Chapter 3: In the last chapter of our dissertation we present the experimental setup, evaluation metrics employed to measure the performance of our models, and performance analysis of the models we used. We conclude this chapter by presenting our authorship attribution interface.

In the end we summarize this dissertation with a general conclusion highlighting the results that we achieved, and also reflecting on the limitations of our models while offering insight into future perspectives.

Chapter 1

Authorship Attribution And State Of The Art

1.1 Introduction

Natural language processing (NLP) is a fascinating field of computer science that combines linguistics, and artificial intelligence in order to help computers comprehend human language. It has become a crucial instrument for processing and analyzing text data as a result of the explosion in digital communication and the enormous amounts of data produced.

Authorship Analysis is one the most intriguing applications of (NLP), which entails identifying the author of a specific text. One of its subfields is Authorship Attribution, which has been an interest of scholars for centuries but it has only recently become feasible to analyze huge amounts of text data and determine authorship with a high level of accuracy thanks to the widespread of deep learning techniques.

In this chapter we will be delving into the various applications of authorship analysis and focus more on the task of authorship attribution that is our field of study, we will also present related works conducted and the results achieved.

1.2 Natural Language Processing (NLP)

Natural language processing is a branch of artificial intelligence that allows computers to process and manipulate the same languages that humans use in written or oral communication to perform a desired task [Coughlin, 1990].

NLP is a multidisciplinary field that incorporates computational linguistics, computing science, cognitive science, and artificial intelligence [Deng and Liu, 2018]. Machine translation, natural language text processing and summarization, speech recognition, artificial intelligence and expert systems, among other fields of study, are a few examples of applications of NLP [Chowdhury, 2003].

1.3 Authorship Analysis

Authorship Analysis is a subfield of natural language processing and forensic linguistics, that is concerned with the evaluation of documents to determine their authorship based on stylometry, which is a linguistic field that analyzes literary style using statistics and it serves as the foundation of Authorship Analysis [Abbasi and Chen, 2005].

1.4 Authorsip Analysis Categories

Authorship analysis studies can be classified into three categories according to [Kaur et al., 2019] (figure 1.1): Authorship Attribution which is our task, Authorship Characterization and Authorship Verification.

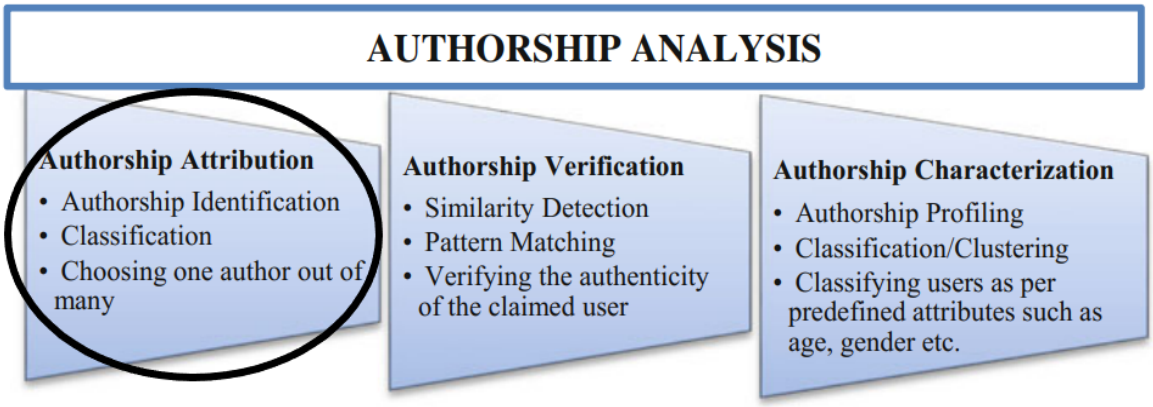

Figure 1.1: Representation of the subfields of Authorship Analysis and their applications [Kaur et al., 2019].

1.4.1 Authorship Attribution

Authorship Attribution (AA) or identification is the task of identifying the author of a disputed anonymous document from among a set of candidate authors for whom written text

samples are available by analyzing the stylistic features of this document. In the literature it is viewed as a text categorization or text classification issue [ELMANARELBOUANANI and KASSOU, 2014], and the first attempts to measure writing style date back to the 19th century with [Mendenhall, 1887] groundbreaking study on Shakespeare's plays, which was followed by statistical studies by [Yule, 1939] and [Zipf, 1932]. Later, the thorough investigation into "The Federalist Papers" authorship by [Mosteller and Wallace, 1963] was unquestionably the most impactful work on authorship attribution [Stamatatos, 2009].

Authorship attribution has practical applications across a wide range of domains, including forensic topics such as detecting plagiarism, identifying the authors of harassment messages, and determining the provenance of bomb letters in counter-terrorism research [Eder et al., 2016].

Authorship attribution approaches can be divided into two types [Stamatatos, 2009]:

Open-set authorship attribution which refers to the case where the real author of the document is not always included in the list of candidate authors [Stamatatos, 2009].

Closed-set authorship attribution on the other hand refers to the situation where the author is included in the candidate set, and the system must determine the true author of the anonymous text [Stamatatos, 2009].

Our work will center on the approach of closed-set authorship attribution. Figure 1.2 illustrates the process of authorship attribution.

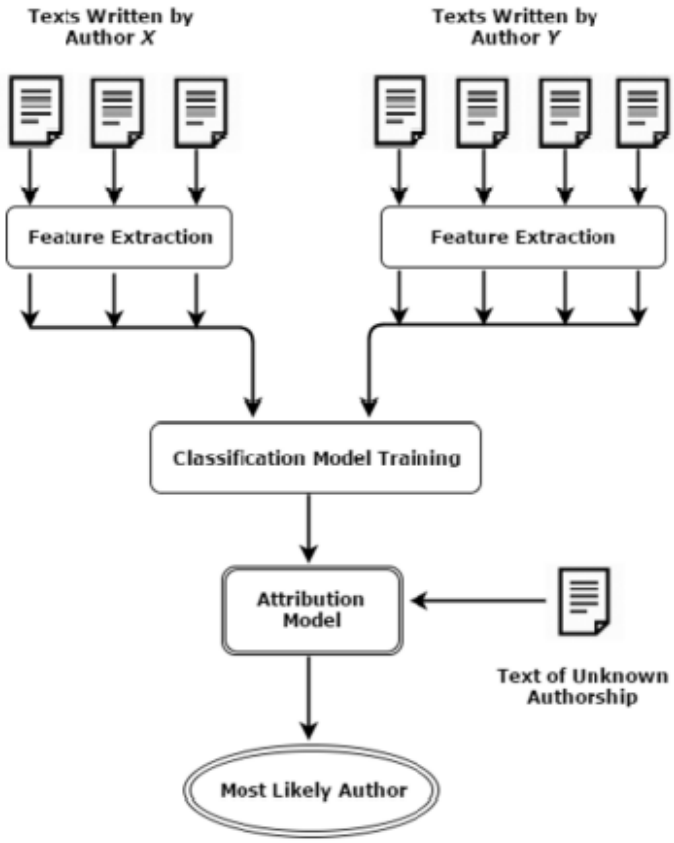

Figure 1.2: Authorship Attribution process [Sharp et al., 2017].

1.4.2 Authorship Characterization

Authorship Characterization or author profiling involves identifying sociolinguistic characteristics such as gender, age, native language, personality type, occupation, and level of education given an anonymous text and a set of predefined categorizations [ELMANARELBOUANANI and KASSOU, 2014]. Author profiling is a topic that has become increasingly significant in various fields such as forensics, security, and marketing [Rangel et al., 2013].

1.4.3 Authorship Verification

Authorship verification refers to the process of determining whether two documents have been written by the same author [Stamatatos, 2016]. This task is considered a significant challenge in authorship attribution, in contrast to authorship attribution models, which seek to identify the most likely author of an anonymous document, an authorship verification model must determine whether the anonymous text is comparable enough to texts provided by a

specific author to conclude that they were created by the same person [Stamatatos, 2016].

1.5 Styolmetric Features

Stylometric features are linguistic or textual characteristics that are extracted from a piece of writing to determine the author or the text's style [Juola, 2008]. When it comes to authorship attribution stylometric features have proven to be a powerful tool in identifying the author of a text and have been widely used in previous works conducted in this field [Stamatatos, 2009]. Stylometric features can be devided to four main categories [ELMANARELBOUANANI and KASSOU, 2014]:

1.5.1 Lexical Features

Lexical Features are used to uncover a person's preferred word and character choice, they can either be word or character based. Word-based lexical features include elements like the overall number of words, the number of words per sentence, the distribution of word lengths, and vocabulary richness, while character-based lexical features include the overall character count, the number of characters per sentence, the number of characters per word, and the frequency of use of specific letters [Abbasi and Chen, 2005].

Lexical features have the advantage that they can be used with any corpus in any language and with only the presence of a tokenizer as extra criteria [Stamatatos, 2009].

1.5.2 Syntactic Features

Syntactic Features refer to the patterns used to construct sentences, this feature category includes tools used to structure sentences, such as sentence length, sentence complexity, punctuation and function words [Abbasi and Chen, 2005].

1.5.3 Structural Features

Structural Features describe the organizational and hierarchical structure of a text, such features include word patterns such as welcomes and signatures, as well as the number of paragraphs and average paragraph length. Structural features have proven to be particularly useful in assessing online messages as they are can help identify patterns and characteristics that are unique to a particular author's writing style [Abbasi and Chen, 2005].

1.5.4 Content-specific Features

Content-specific are words that are relevant within a certain topic domain [Abbasi and Chen, 2005], they can be used to analyze the vocabulary and topic of a text and can be particularly useful in authorship attribution tasks where the texts being analyzed are focused on a specific content area.

1.6 Related Works

In this section we present the related works conducted in the field of Authorship Attribution, the related works can be classified according to the methods used i.e.,machine learning or deep learning, as well as the dataset in question whether it is from social media or other sources.

1.6.1 Machine Learning Approaches In Social Media

For Arabic authorship attribution of short texts and more precisely tweets [Kah et al., 2022] propose using various stylometric features to quantify the writing style of authors, three types of n-gram models are also used in addition to using both TF-IDF and count vectorizer for feature extraction, for the final classification they used three models namely the Naïve Bayes classifier, Support vector machines, and Random forests, which are first implemented individually then later combined to build a bagging classifier which provides the final prediction based on a majority voting process. They used a dataset composed of 71,391 Arabic tweets posted by 44 different authors, they also examined the impact of various factors on the accuracy of authorship attribution, including the size of the training dataset, the number of classes covered, the text processing techniques applied, the methods used for feature selection and extraction, and the classifier implemented. The results show that the highest accuracy achieved was 97.4% which is among the best results reached so far in arabic authorship attribution of short texts.

[Theophilo et al., 2022] propose an extension to the LIME (local interpretable model-agnostic explanations) technique to improve the explanations of the state-of-the-art methods for authorship attribution on social media posts, they suggest incorporating character n-grams into LIME as interpretable components.They use a dataset composed of 130 million messages from more than 56,000 Twitter users.

[Rabab'ah et al., 2016] attempt to solve the authorship attribution problem of arabic tweets by combining different feature sets namely stylometric features, MADAMIRA Features, and Bag-Of-Words. They used three different classifiers specifically Naïve Bayes classifier, Support

Vector Machine, and Decision Tree. To evaluate the models proposed they use a dataset containing 38,386 tweets from 12 different users, the results demonstrate that combining all feature sets gives the best results, the highest accuracy achieved was 68.67%, which was reached by using the SVM classifier on the combined feature sets.

1.6.2 Machine Learning Approaches In Other Sources

For the PAN at CLEF 2019 authorship attribution task [Muttenthaler et al., 2019] developed three distinct n-gram models, a variable-length n-gram model was created for each of the three models. They implemented a word n-gram model (1-3 gram), a distorted character n-gram model (1-3 gram) and a standard character n-gram model (2-5 gram). Then an SVM classifier was developed for each of the three models. They used the fanfiction dataset from the 2019 PAN at CLEF competition, according to the results the standard character n-gram model performed best and achieved an F1-score of 69%.

To address the Authorship Attribution in brief texts problem [Zhang et al., 2018] propose a new model Author-document topic model (ADT) which develops the model for the corpus at the document level as well as the author level. Additionally, they suggest a new classification algorithm to determine text similarity in order to identify the authors of anonymous texts. For the experimentation with their model, they use two datasets the first one is Pan'11 emails which contains 11936 emails with 72 authors and the second one is Blog which contains 678161 blogs with 19320 authors. The accuracy obtained by their model on the Pan dataset reaches an accuracy of 54.7% and 49.2% in the Blog dataset.

1.6.3 Deep Learning Approaches In Social Media

To address the issue of Author Attribution of brief texts [Suman et al., 2022] proposed using a capsule-based CNN model as well as using a capsule based kervolutional neural network (KNN), and BERT embeddings for text representation as well as providing a thourough comparaison of performance of different text representations, including GloVe embedding, BERT embedding, character unigram, and character bigram. They evaluate the performance of their developed systems using the [Schwartz et al., 2013] dataset consisting of 9000 Twitter users and 1000 post each, totaling 9 million posts, from which they randomly selected 50 users with 1000 tweets each. The results show that the highest accuracy was obtained by the capsule-based CNN using character unigrams which reached 86.62%.

[Wang and Iwaihara, 2021] suggest a hybrid model that is composed of two main parts, the

first part is a pretrained language model based on RoBERTa to create post representations that are aware of tweet-related stylistic features and their contextualities, the second section is a CNN model constructed using a number of feature embeddings to represent users' writing styles they then put these representations together for the final AA classification. They evaluate their method on the [Schwartz et al., 2013] dataset, their best performing model achieved an accuracy of 88.2%.

[Huang et al., 2020] applied an enhanced character embedding technique to typical neural networks specifically CNN and LSTM, A mixed sequence of word and character n-grams is input to their suggested architecture, they also focus on the textual implicit characteristics and incorporate ten latent posting styles into the models such as text length, user mentions, and sentiment orientation. They then obtained a compact feature vector representation by automatically extracting textual features from the sequence using neural network models. In order to identify the author, the representation is finally passed to a fully connected module with the softmax function. They evaluate the performance of their model using the dataset of [Schwartz et al., 2013] , their best performing model which was CNN using mixed words combined with character n-grams and latent posting styles achieved an accuracy of 83.6% .

To solve the problem of Authorship Attribution in short texts [Shrestha et al., 2017] proposed an architecture that is a CNN that uses a sequence of character n-grams as input and is able to learn to learn the representation of the text starting from the character sequence. They use the [Schwartz et al., 2013] dataset with variating number of authors and tweets for the evaluation of their models. Their best performing models being CNN-1 and CNN-2 obtain the best results with 50 authors and 1000 tweets each with an accuracy of 75.7% and 76.1% respectively for both models.

1.6.4 Deep Learning Approaches In Other Sources

[Lam et al., 2021]'s method consisted of using two CNN architectures that they note as CNN-1 and CNN-2 as well as 4 LSTM models two of them being unidirectional LSTM-1 and LSTM-2 and the other two bidirectional BiLSTM-1 and BiLSTM-2. The dataset used in this work is that of the 2017 French Elections related tweets it consists of 42923 tweets in total and 11 candidates. The results of the training prove that both CNN's and LSTM's perform well with CNN's slightly outperforming LSTM's, the CNN-2 model reached the best performance out of all the models with an accuracy of 83%, the results also show that bidirectional LSTM's have better performance when being compared to unidirectional LSTM's.

[Fabien et al., 2020] introduce BertAA, a fine-tuning of a pre-trained BERT language model

that does authorship classification using an additional dense layer and a softmax activation, Additionally, the authors assess the effects of adding extra features, such as stylometric and hybrid features.They used different corpora for the task including: Enron Email corpus which contains 130000 emails, the IMDb Authorship Attribution corpus which contains 271000 movie reviews produced by 22116 authors also the IMDb62 corpus containing 62 authors and 1000 text per author, and lastly the Blog Authorship Attribution Corpus which contains 680000 posts from approximately 19000 authors, the accuracy achieved by their model using 100 authors is 97% in the Enron dataset, 86.1% on the IMDb dataset, and 58.8% on the Blong dataset.

[Sari et al., 2017] proposed using continuous n-grams representations with a feed forward neural network they also proposed three variations for the continuous n-grams: continuous word n-grams, continuous character n-grams, and combining both character and word n-grams. They test their models on four different datasets, the first one is the Judgement dataset which was collected from judgment writing of three Australian High Court's judges, the second is subset of Reuters Corpus Volume 1 and includes newswire articles written by 10 contributors noted CCAT10, the third one is an expanded variation of the previous one there are 5,000 documents overall from 50 writers noted CCAT50, and lastly the fourth dataset consists of 62,000 movie reviews and 17,550 message board posts written by 62 active users of the Internet Movie Database (IMDb) noted IMDb62. They presented results of the three different models across the four datasets, the best accuracy obtained was by the Continuous n-gram char (2,3,4) model on the IMDb62 dataset as the accuracy reached 94.80%.

Table 1.1 illustrates a summary of the related works and their results.

Table 1.1: Summary of the related works and their results

Article	Approach	Model	Dataset	Results
Suman et al., 2022	Deep Learning	KNN,CNN	Schwartz et al.,2013	Accuracy: 1-CNN: 86.62% 2-KNN: 83.82%
Wang and Iwaihara, 2021	Deep Learning	RoBerta, CNN	Schwartz et al.,2013	Accuracy: 88.2%
Lam et al., 2021	Deep Learning	CNN, LSTM, BiLSTM	2017 French Elections: 42923 Tweets 11 Users	Accuracy: CNN-2: 83% BiLSTM-2: 81% F1-score: CNN-2:83% BiLSTM-2:80% Precision: CNN-2:82% BiLSTM-2:80% Recall: CNN-2: 82% BiLSTM-2:97%
Huang et al., 2020	Deep Learning	CNN, LSTM	Schwartz et al.,2013	Accuracy: 1-CNN: 83.6% 2-LSTM: 76.21
Fabien et al., 2020	Deep Learning	BertAA with an additional dense layer and softmax activation	Enron Email corpus, IMDb, IMDb62	Accuracy: 1-Enron: 97% 2-IMDb: 86.1% 3-Blog: 58.8%
Shrestha et al., 2017	Deep Learning	CNN	Schwartz et al.,2013	Accuracy: 1-CNN-1:75.7% 2-CNN-2: 76.1%
Sari et al., 2017	Deep Learning	FFNN	CCAT10, CCAT50, IMDb62, Judgement	Accuracy: Continuous n-gram char(2,3,4): 94.80%
Kah et al., 2022	Machine Learning	NB, SVM, RF, Bagging	71,391 Tweets 44 Users	Accuracy: Bagging: 97.4%
Theophilo et al., 2022	Machine Learning	LIME	130 million Tweets 56000 Users	-
Muttenthaler et al., 2019	Machine Learning	SVM	2019 PAN at CLEF Fanfiction dataset	F1-score: 69%
Zhang et al., 2018	Machine Learning	ADT	Pan'11 emails, Blog	Accuracy: Pan'11 emails: 54.7% Blog: 49.2%
Rabab et al., 2016	Machine Learning	NB, SVM, DT	38,386 Tweets 12 Users	Accuracy: SVM: 68.67%

Table 1.1 summarizes the different methods used in the related works as well as their influence on the results, the table sheds light on the performance of deep learning and machine learning methods, it can be observed that deep learning methods perform significantly better than machine learning methods, although the highest accuracy achieved was through the utilization of machine learning. It is crucial to acknowledge that this result was attained by utilizing a relatively small dataset consisting of only five users, with 500 tweets each. It can also be observed that character and also word n-grams are employed in the majority of the related works for analyzing and distinguishing writing styles among different authors. Among the various deep learning methods, CNN's and LSTM's stand out as the most extensively used in most of the related works in authorship attribution thanks to their effectiveness in capturing patterns and features in textual data. The experiments conducted in each of the related works also reveal that the size of the dataset matters and increasing the number of authors and decreasing the number of texts negatively impacts the performance of the models. Figure 3.3 illustrates the division in the related works.

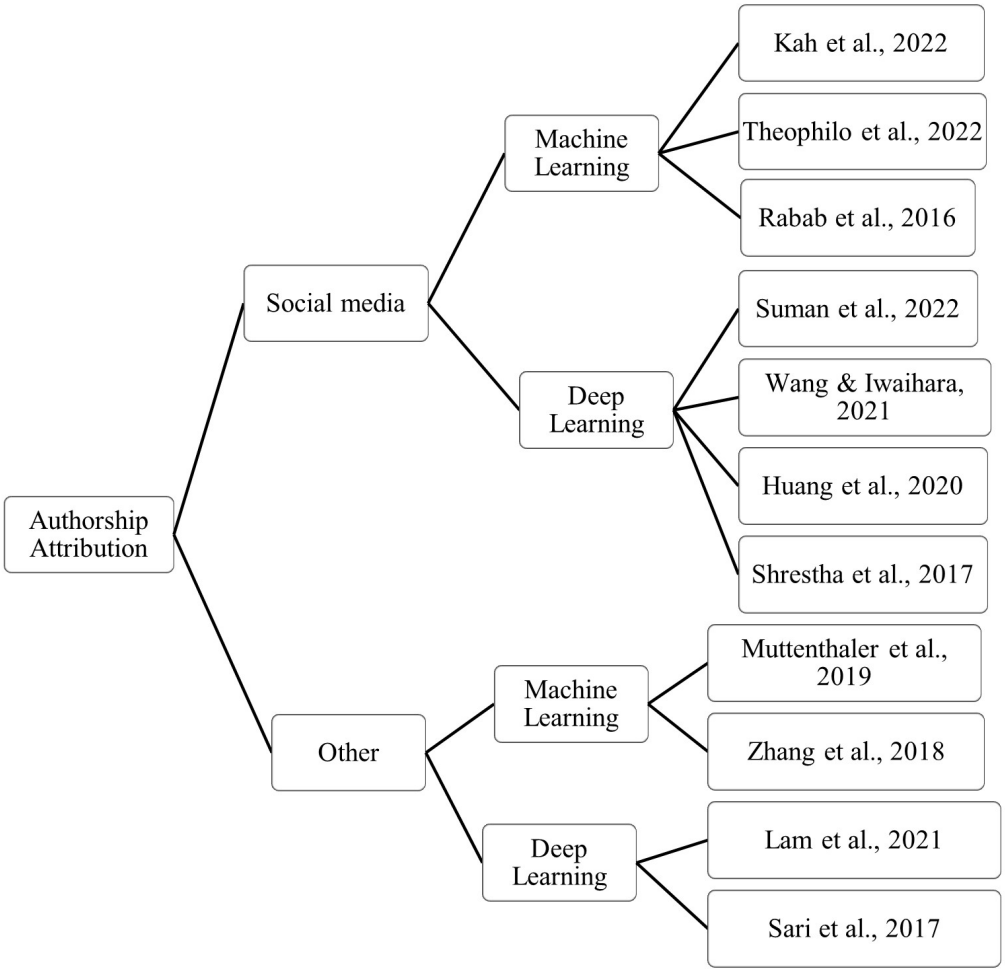

Figure 1.3: Representation of the division in the related works.

1.7 Conclusion

In this chapter we have presented the field of Authorship Analysis as well as its different subfields (Authorship Attribution, Authorship Verification, Authorhsip Characterization) and focused precisely on Authorship attribution which represents our task, Later on we presented and explained the different types of stylistic features used in Authorship Attribution we have also presented different related works conducted and the methods that have been used to solve the AA problem in social media as well as other sources.

In the next chapter we will present our proposed approach and the methods we used.

Chapter 2

Proposed Solution

2.1 Introduction

Traditionally authorship attribution has been performed using stylometric features such as lexical or syntactic features. However, recent advances in NLP as well as machine learning and deep learning have led to the development of more advanced techniques for authorship identification which have been used by the previous works and have demonstrated great potential in enhancing the results of this task.

In this chapter we will present in detail our proposed architecture to tackle the problem of authorship attribution in short texts, we will also be describing the dataset that has been used and the different steps necessary to prepare our data.

2.2 Overall Architecture

As a solution to the Authorship attribution of brief texts problem we adopted an architecture that integrates the following stages (figure 2.1):

1-Data Pre-processing: Where the unstructured raw text is cleaned, standardized, and transformed into a structured format.

2-Vectorization: Wherein the processed text is converted into numerical representations, effectively capturing the essence of linguistic patterns and style in order to be processed by the proposed models.

3-Train/Test/Validation split: This step includes splitting the data into three distinctive sets: the train, test, and validation sets.

4-Model training: In this step we feed the vectorized data into our proposed models in order to train them.

5- Model evaluation: Lastly in this stage we evaluate our models with new unseen data (test data) to effectively asses their performances.

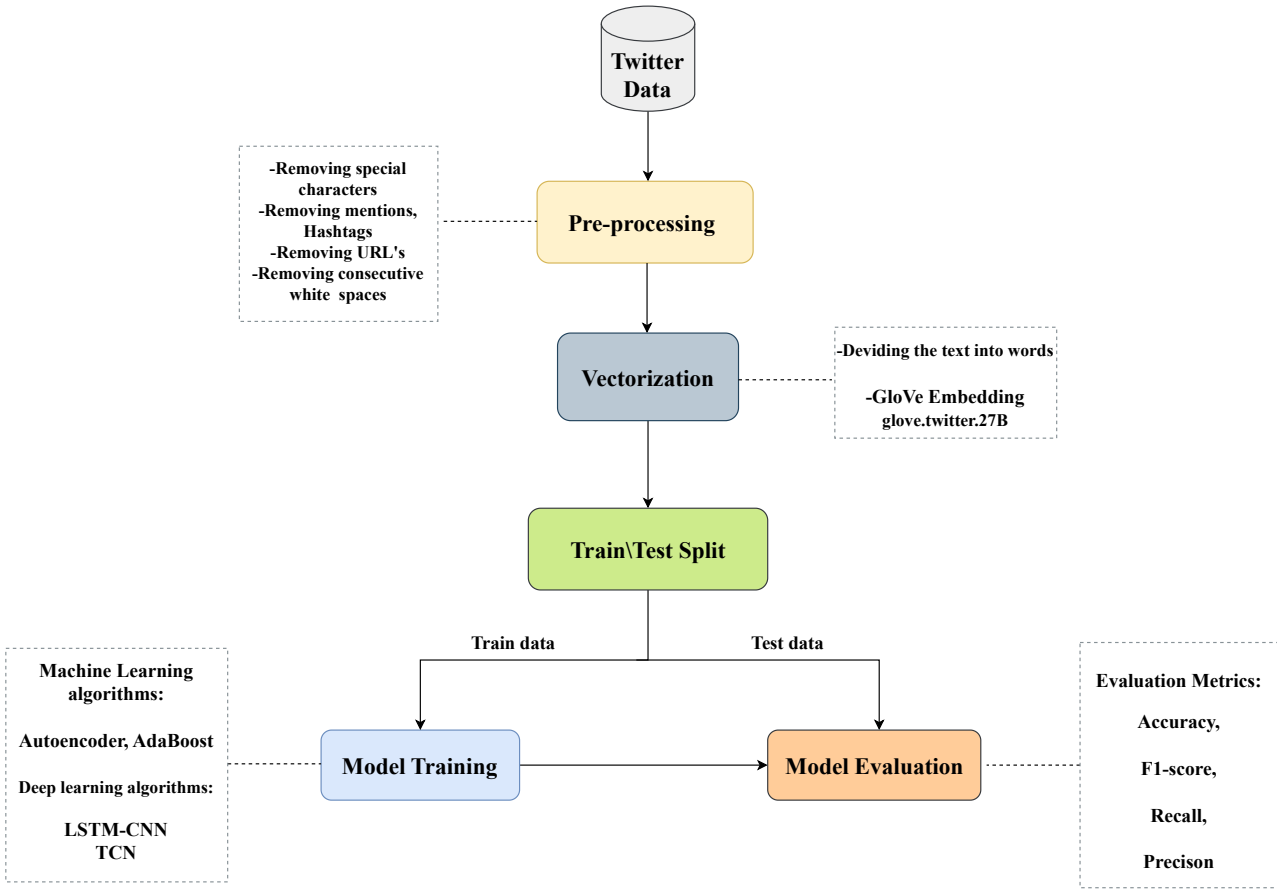

Figure 2.1: Global scheme of our proposed approach.

2.3 Dataset

In authorship attribution datasets are essential since they serve as the foundation for the creation and evaluation of models that can accurately determine the authorship of a given text, because Twitter has such a large and diverse collection of tweets from users worldwide it has become a well known source of data for various tasks in NLP. For our authorship attribution of

short texts problem we chose to use the [Schwartz et al., 2013] Twitter dataset which has been widely used by researchers in the authorship attribution task. It consists of overall 7026 users and 1000 tweet for each user, which makes it a total of approximately 7 million tweets. The dataset provided by [Schwartz et al., 2013] originally contained two main folders described as follows (table 2.1):

Varying_number_of_authors: This folder contains 1 789 folders each of them representing one user and each folder containing 10 bz2 files that contain 20 tweets each.

Varying_training_set_size: This folder contains 7026 user folders with 100 tweet in each bz2 file totaling 1000 tweet per user.

After reviewing the relevant literature in our research area, we chose to experiment with two different sizes of the dataset, the first one with randomly choosing 50 users with 1000 tweets each, and the second one using 10 authors with 1000 tweets each.

Table 2.1: Description of the dataset

Folder Name	Subfolder Name	Subfolder Count	Tweet Count
varying_training_set_size	User_id	7026	7026x10x100
varying_number_of_authors	User_id	1789	1789x10x20

2.4 Pre-processing

Twitter data is usually very unstructured, this step aims to transform twitter data into a standardized format that is easier to analyze and to remove noise present in the data, below is a pseudo code describing the steps we followed for data cleaning.

Algorithm 1 Data pre-processing algorithm

```
1: procedure PreprocessTweets(Twitter_data)
2:     INPUT: Twitter_data
3:     OUTPUT: Clean_tweets
4:     x ← Remove_URL(Twitter_data)
5:     x ← Remove_mentions(x)
6:     x ← Remove_Hashtags(x)
7:     x ← Remove_special_characters(x)
8:     x ← Remove_consecutive_spaces(x)
9:     clean_tweets ← Lowercase(x)
10: end procedure
```

Removing URLs: This step removes urls.

Removing mentions and hashtags: Mentions denoted by the '@' symbol are references to other users, removing them helps alleviate noise within the text, similarly hashtags which are denoted by the '#' symbol are used to index keywords or topics on Twitter, by eliminating these elements the focus shifts more on the actual text.

Removing special characters: This step eliminates special characters including punctuation marks, emojis, and other non-alphanumeric symbols, in order to reduce noise and further simplify the text.

Removing extra spaces: This step aims to remove consecutive white spaces and replace them with a single white space.

Converting the text to lower case: In this step we converts all text to lowercase to reduce the number of unique tokens in the data and help avoid case-sensitivity issues.

Table 2.2 demonstrates the transformations of this text : "@AMPlifiedPhotos Smh! But ty this site if you didn't get one yet http://www.watch-movies-online.tv/ <-that one :)" after each pre-processing previously mentioned.

Table 2.2: Representation of the Pre-processing steps applied to text

Pre-processing step	Pre-processed text
Removing URLs	@AMPlifiedPhotos Smh! But ty this site if you didn't get one yet ~~http://www.watch-movies-online.tv/~~ <-that one :)
Removing mentions and hashtags	~~@AMPlifiedPhotos~~Smh! But ty this site if you didn't get one yet <-that one :)
Removing punctuation and special symbols	Smh ! But ty this site if you didn't get one yet ~~<-~~ that one
Removing extra white spaces	Smh But ty this site if you didn't get one yet that one
Converting the text to lower case	smh but ty this site if you didn't get one yet that one

2.5 Vectorization

Vectorization refers to the operation of converting text data into numerical vectors, since computers can't understand text data, vectorization is an essential step as it transforms unstructured data into a format that captures semantic meaning, relationships, and patterns.

Before converting our data into numerical representations we must first go through the tokenization step followed by the padding, both these steps are described in the next sections.

2.5.1 Tokenization

Tokenization is the process of dividing our text into smaller units or more commonly known as tokens such as words, phrases, or even characters. [Bird et al., 2009] It is an important step to include before vectorization as it helps process and extract valuable information from our textual data. Our text will be tokenized into words.

2.5.2 Padding

Padding is the step where extra values are added to the beginning or end of our data sequences to guarantee that they have consistent dimensions, the shortest sequences are expanded to match the length of the longest sequences in the dataset by adding a specific padding value which is often zero. There are two types of padding : pre-padding which involves adding values to the beginning of our sequences, and post-padding which is where we add values at the end, in our architecture we opted for the post-padding option.

2.5.3 Word Embedding

Word embedding is a method used to map textual words or phrases into a low-dimensional continuous space, this mapping makes it possible for similar words to have similar encoding [Birunda and Devi, 2021]. Word embedding is a powerful tool widely used in various modern natural language processing tasks including text classification, document clustering, and sentiment classification [Birunda and Devi, 2021]. In our work, we will be using GloVe embeddings for word representation, the detailed explanation of GloVe embedding will be provided below.

2.5.3.1 Glove

GloVe (Global Vectors) is a word embedding model proposed by [Pennington et al., 2014], it is based on word occurrences in a corpus of text, the primary difference between GloVe and any other word embedding model such as word2vec is that it considers more than just the immediate context of each word, it also considers the overall distribution of words across the entire corpus by using a co-occurrence matrix of words where Xi j is the frequency of the word i co-occurring with the word j [Naili et al., 2017].

For our task we chose to use the Glove embedding method because it has pre-trained models available on several different corpora, including Twitter which is more relevant to our task considering our Twitter dataset. Algorithm 2 shows the functionality of the GloVe embedding.

Algorithm 2 GloVe embedding algorithm

```
1: procedure GloVeEmbedding(tokenized_tweets)
2:     INPUT tokenized_tweets
3:     OUTPUT tweet_embeddings
4:     co_occurence_matrix ← []
5:     context_words ← []
6:     word_vectors ← []
7:     for each tweet in tokenized_tweets do
8:         for each word in tweet do
9:             context_words ← get_context_words(word)
10:            for each context_word in context_words do
11:                co_occurence_matrix ← update(co_occurence_matrix, word, context_word)
12:            end for
13:        end for
14:    end for
15:    for each word in co_occurence_matrix do
16:        word_vectors[word] ← initialize_random()
17:    end for
18:    for each word in co_occurence_matrix do
19:        for each context_word in co_occurence_matrix[word] do
20:            word_vectors[word] ← gradient_descent_update(word_vectors)
21:        end for
22:    end for
23: end procedure
```

2.6 The Train/Test/Validation split

In order to examine how well the trained model performs on new unseen data and access its ability to generalize we must split our data into three subsets each used for specific tasks described below:

The train set: It represents the largest portion of the dataset and it is used to train the model by providing input data and the labels that correspond to it, the model learns the relationships in the data and optimizes its weights and parameters.

The test set: This set is used to evaluate our model's performance on new data and its ability to make predictions on data that is separate from the training set.

The validation set: The validation set's purpose is to estimate our model's performance during the training phase and is useful for tuning the hyperparameters, it is also useful in detecting when our model is overfitting.

For our task we accorded 70% of the data to the training set, 20% to the test set, and 10% for the validation set.

2.7 Models

In this section we will present the models we have used to solve the authorship attribution in brief texts problem, researchers in this field have used various methods and architectures namely Convolutional Neural Networks (CNN) which have proved to have a very good performance, as well as Long Short-Term Memory Networks (LSTM) which have also been widely used in previous works.

For our approach we have used three different architectures: Temporal Convolutional Network (TCN), the combination of Long Short-Term Memory network (LSTM) and Convolutional Neural Network (CNN), and an Autoencoder model combined with AdaBoost Classifier. Each one of these architectures will be explained in detail in the sections that follow.

2.7.1 Temporal Convolutional Network (TCN)

The first architecture that we propose is a Temporal Convolutional Network (TCN). TCN's are a type of deep learning model that are designed for the purpose of handling sequential data or what is known as time series. First proposed by [Bai et al., 2018] they were developed as an alternative to Recurrent Neural Networks for modeling temporal dependencies in data.

The TCN's advantage like RNN's is that they can efficiently handle inputs of varying length, they have the ability to take in a sequence of any length and map it to an output of the same length, secondly the convolutions in the architecture are causal, meaning that there is no information leakage from future to past [Bai et al., 2018]. They are also more memory efficient than RNN's thanks to the shared convolution architecture which enables them to process lengthy sequences in parallel [Lara-Benítez et al., 2020].

Temporal Convolutional Networks consist of three major parts [Zhu et al., 2022]:

2.7.1.1 Causal Convolutions

Causal convolutions are a particular type of convolutions that ensure that the result at any given time step depends only on prior time steps and not on future time steps, in a more technical manner: output at time t is convolved only with elements from time t and earlier in the previous layer [Bai et al., 2018] (figure 2.2).

2.7.1.2 Dilated Convolutions

The use of dilated convolutions is a key feature of TCN, filters slide over adjacent elements in the input sequence in traditional convolutions. In dilated convolutions, however, the filters skip a certain number of elements (figure 2.2) which is represented by the dilation rate between each application, thus, dilation translates to inserting a fixed step between every two adjacent filter taps [Bai et al., 2018].

Employing a dilated convolution enables the model to have an exponentially large receptive field, using a larger dilation also allows a top-level output to represent a broader range of inputs [Bai et al., 2018].

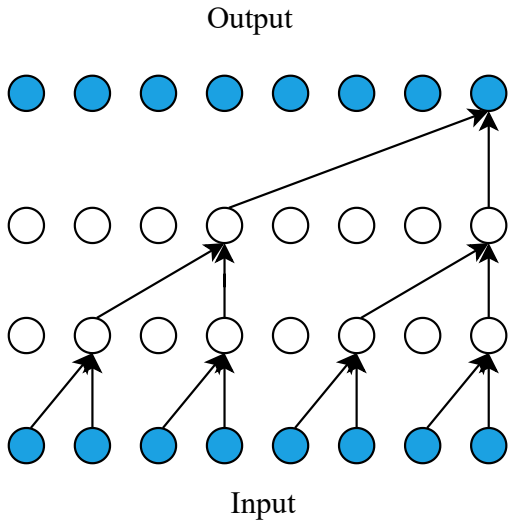

Figure 2.2: Representation of dilated causal convolutions with dilation rates [1,2,4].

2.7.1.3 Residual Connections

Residual Connections are an important component in TCN that was proposed to overcome the vanishing gradient problem and to make training a deep TCN architecture easier [Zhu et al., 2022]. They consist of adding the input of a TCN block to its output, the use of residual blocks thus allows to pass original information from the input directly to the deeper layers [Zhu et al., 2022].

Within a residual bloc as illustrated in figure 2.3, the TCN has two layers of dilated causal convolution and non-linearity which was achieved using a ReLU activation, a weight normalization is applied to normalize the convolutional filters, and for regularization a spatial dropout

is implemented after each dilated convolution, an additional 1x1 convolution is also employed to ensure that element wise addition receives tensors of the same width as the input-output widths [Bai et al., 2018].

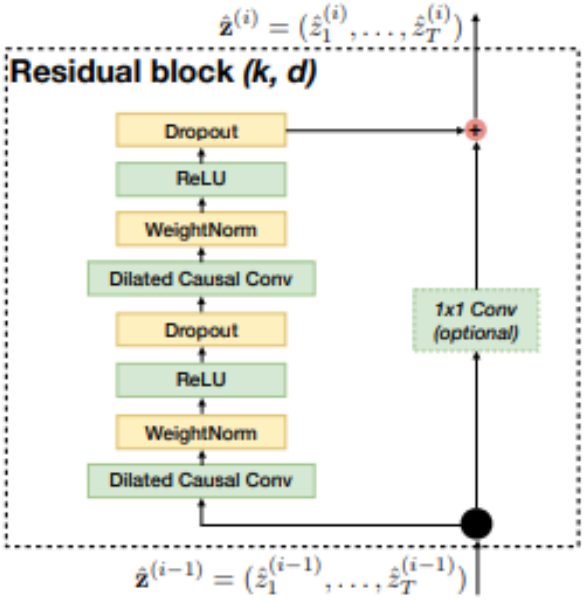

Figure 2.3: Illustration of a TCN Residual Bloc [Bai et al., 2018].

2.7.2 Proposed TCN Architecture

For our first architecture we have chosen to work with TCN because it has shown competitive performance in a variety of sequence-based tasks, such as language modeling and sentiment analysis and for its ability to capture long-range dependencies in sequential data. And when it comes to authorship attribution where identifying the unique writing styles of an author requires considering the entire text, the TCN model seems to be suitable for our task.

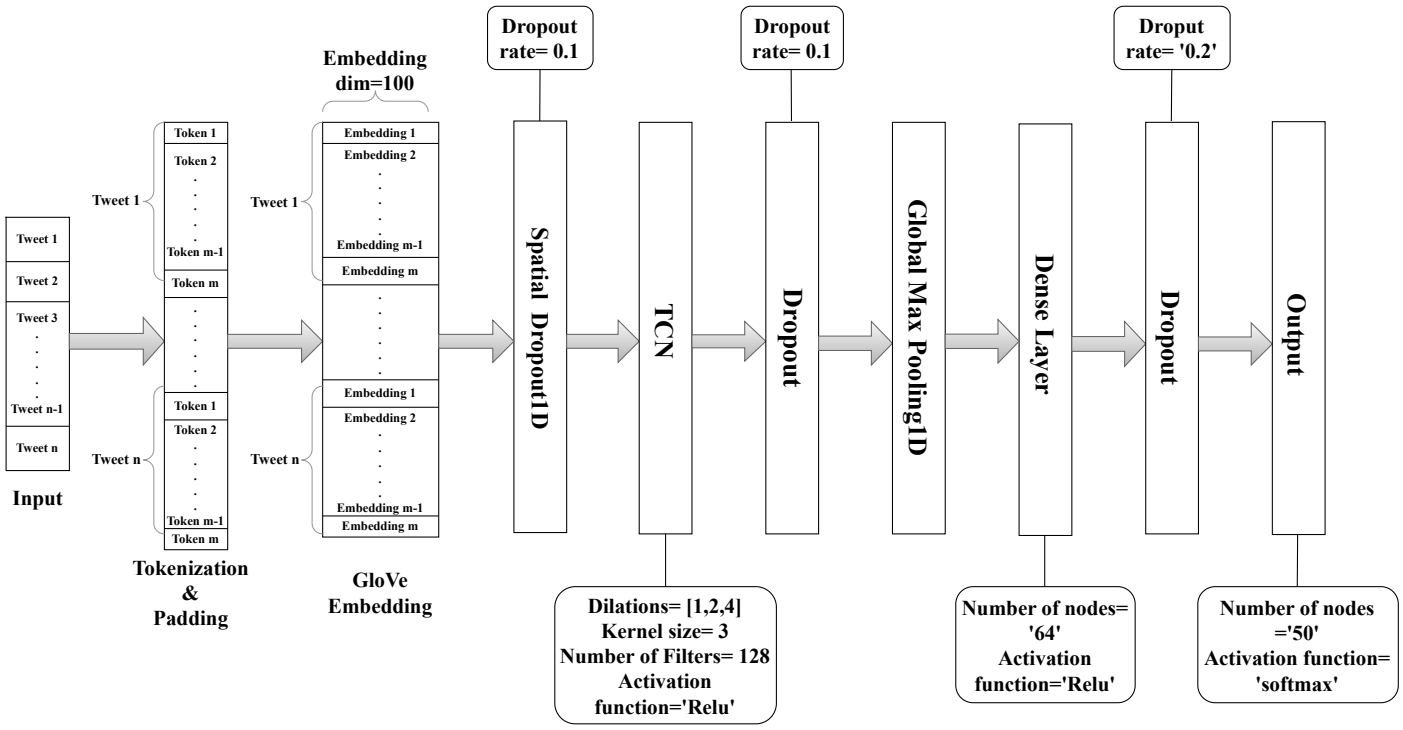

Figure 2.4: TCN Architecture for Authorship Attribution

Figure 2.4 represents our proposed TCN architecture, and we further explain it as follows:

- The input which takes in the pre-processed then tokenized and padded tweets with a max length equal to 100.
- Next an embedding layer using the variant of GloVe that was pretrained on large amounts of twitter data is applied, the purpose of this layer is to convert our input data into continuous vector representations that can be easily processed by our model as they can capture relationships between the different words in the input. We use 'glove.twitter.27B' and an embedding dimension equal to a 100.
- Next we use a Spatial Dropout which is a regularization technique used in CNN's that prevents overfitting by randomly setting some input units to zero during training, we use a rate equal to 10%.
- A TCN bloc is next applied to extract meaningful features that are present in our data, our TCN uses causal convolutions with dilation rates 1,2, and 4 with 128 filters and a kernel size of 3, and a 'ReLu' activation function.
- After the TCN bloc we apply a dropout with a 10% rate to help prevent overfitting and make the model generalize better.

- GlobalMax Pooling 1D is applied which is a pooling operation that minimizes the dimensions of the feature maps by taking the maximum value of each feature map and assembling them to create a new vector, we use it for reducing the feature maps prodcued by the previous TCN layer.
- We apply a fully connected layer next with 64 nodes and a 'Relu' activation.
- Finally, we have the output layer with 50 nodes which correspond to the number of our classes with a Softmax activation.

2.7.3 Long Short Term Memory Networks (LSTM)

Long Short-Term Memory networks also known as LSTM's are a type of neural network that emerged as a solution to the vanishing gradient problem and an enhancement to the recurent neural networks, they have proven to be the best performing technique in speech and language processing and improve handling of long-term dependencies in sequential data [Arras et al., 2019].

A typical LSTM unit is made up of a cell and three types of gates: input gates, output gates (figure 2.5), and forget gates [Onan, 2022], we will explore each of these concepts in the next sections.

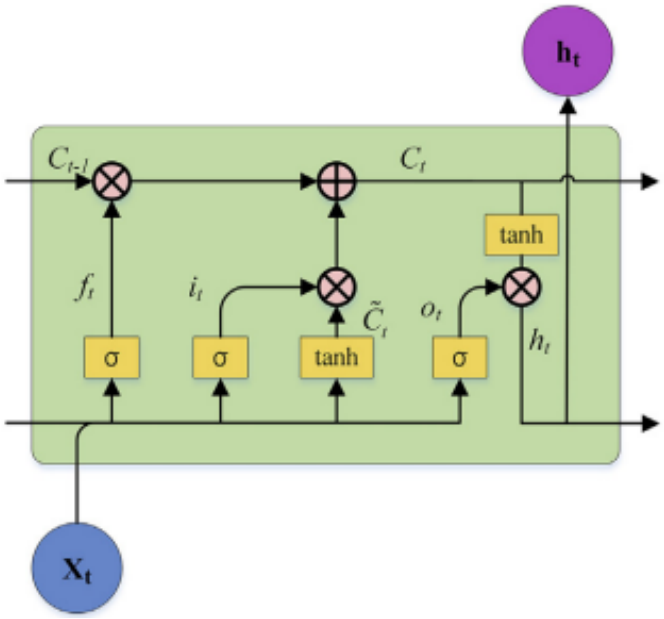

Figure 2.5: LSTM unit [Li et al., 2017].

2.7.3.1 Memory Cells

The memory cells are the central processing and storage unit in an LSTM [Arras et al., 2019], the cells determine which information should be stored and when units should access the information based on gate open and close operations [Onan, 2022]

2.7.3.2 Input Gate

The input gate determines how much of the incoming information should be stored in the memory cell [Bisong, 2019]. It takes the current input and the previous hidden state as inputs and passes them through a sigmoid activation function [Arras et al., 2019]. The output of the sigmoid function is then multiplied by a candidate activation vector, which represents the new information to be added to the memory cell.

2.7.3.3 Forget Gate

The forget gate controls how much data from the long-term state is stored across time instants [Bisong, 2019]. It takes the current input and the previous hidden state as inputs and passes them through a sigmoid activation function. The output of the sigmoid function is multiplied by the previous memory cell state, determining which information should be forgotten [Onan, 2022].

2.7.3.4 Output Gate

This gate controls how much information to output from the cell at a particular time instant [Bisong, 2019]. It takes into account the current input, the previous hidden state, and the current memory cell state, and decides how much of the information should be passed through to the output [Goodfellow et al., 2016].

2.7.4 Convolutional Neural Networks (CNN)

Convolutional Neural Networks or CNNs are a type of neural network that has a known, grid-like structure for processing data their name indicates that the network employs a mathematical operation called convolution, which is a specific kind of linear operation [Goodfellow et al., 2016].

CNNs have been immensely successful in many computer vision tasks such as image classification but they have also been used in other areas such as speech recognition and natural

language processing [Salvaris et al., 2018]. The two main layers in a CNN are the convolutional layer, and the pooling layer, we will explain these layers below.

2.7.4.1 Convolutional layers

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction which consists of combining a convolution operation and an activation function [Yamashita et al., 2018].

The convolution operation is a kind of linear function used for feature extraction it applies what is known as a kernel or a filter (which is a tensor with specific size), across the input to produce the output which is called a feature map [Yamashita et al., 2018]. A convolutional layer can have multiple kernels used to extract different types of features [Bisong, 2019].

When building a convolutional layer, important factors to keep in mind are [Bisong, 2019]: the filter size, the stride of the filter which determines the step size at which the filter moves across the input data during the convolution operation, and the padding of the input layer which is a technique to maintain spatial information at the input's edges where extra border pixels(typically zeros) are added to the input data before applying the convolution operation [Bisong, 2019].

2.7.4.2 Pooling layers

Pooling layers are layers that typically follow one or more convolutions, the purpose of the pooling layer is to downsample or reduce the feature maps produced by the convolutional layer [Bisong, 2019]. It is possible to think of the pooling layer as an aggregation function that gathers previously learnt features and extracts the essential features from previous layers [Bisong, 2019]. The pooling layer performs the aggregation operations that include max, sum, and average, the max is the most commonly used type of aggregation function and it is referred to as Max Pooling [Yamashita et al., 2018].

Similarly to convolutional layers hyperparameters in polling layers include filter size, stride, and padding [Yamashita et al., 2018].

2.7.5 Proposed LSTM-CNN Architecture

The second architecture that we proposed is a combination of LSTM and CNN layers, the LSTM layers have proven to be good at capturing unique writing styles and patters present in

tweets, we use them to extract meaningful features which are then fed into CNN layers which further identify patterns in the data. Finally, the output processes these combined features in order to produce the final prediction.

We visualize this described architecture with the following diagram in figure 2.6.

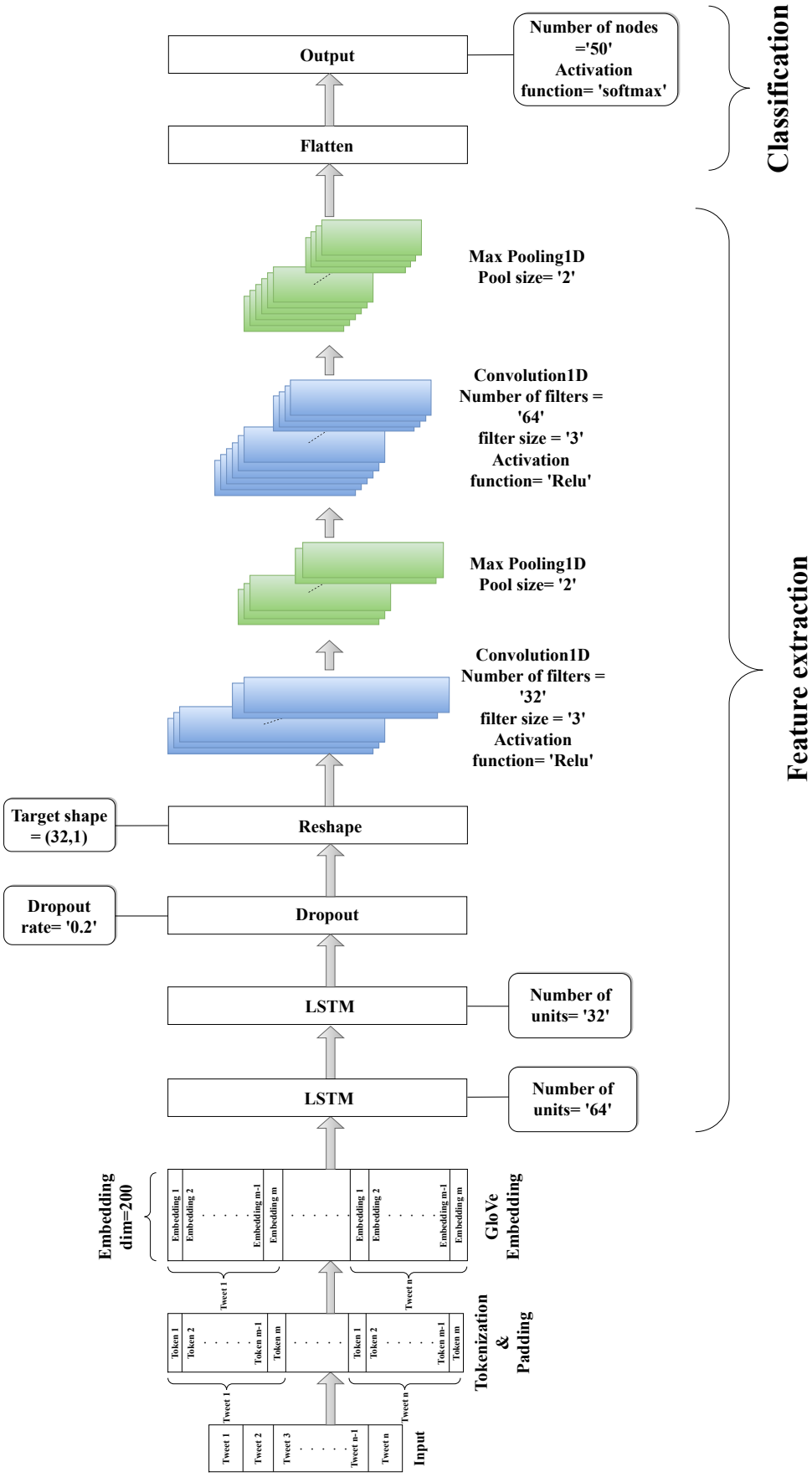

Figure 2.6: Our proposed LSTM-CNN architecture.

The architecture can be further explained along these lines:

- First the input layer receives the pre-processed and tokenized tweets, the max length is equal to 100.
- Next we pass the input to a GloVe embedding layer we use 'glove.twitter.27B' with the embedding dimension equal to 200.
- Following the embedding layer we have the first LSTM layer with a number of units equal to 64, the LSTM layers capture the unique writing styles and extract meaningful features present in the tweets.
- A second LSTM layer comes after with 32 units.
- Then we apply a droput with a rate of 0.2 to avoid overfitting.
- A first 1D convolution layer is later applied with 32 filters and a filter size of 3 and a 'Relu' activation.
- After the convolutional layer we apply a Max pooling layer with a pool size equal to 2 in order to reduce the dimensionality from the convolutional layer's output and keep the most important features.
- Next we have another convolutional layer with 64 filters and a kernel size equal to 3 and 'Relu' activation, followed by another Max pooling layer with a pool size equal to 2.
- We later apply a flatten layer to transform the feature maps produced by the previous convolutional layers into a 1D array in order to be processed by the fully connected layer.
- Finally, we have the output layer with 50 nodes and a 'Softmax' activation in order to produce the final prediction.

2.7.6 Autoencoder

An Autoencoder is a type of neural network that is trained to attempt to copy its input to its output, an autoencoder consists of two main parts: the encoder function which encodes the high-dimensional input into a low-dimensional hidden variable and the decoder function that attempts to produce a reconstruction of the input [Goodfellow et al., 2016].

Autoencoders are designed so that they do not replicate the input data precisely and are constrained in a way that allows them to copy only approximately, they thus learn the most

important features to reconstruct the input data [Salvaris et al., 2018]. Figure 2.7 presents the architecture of an autoencoder using tweets as an example.

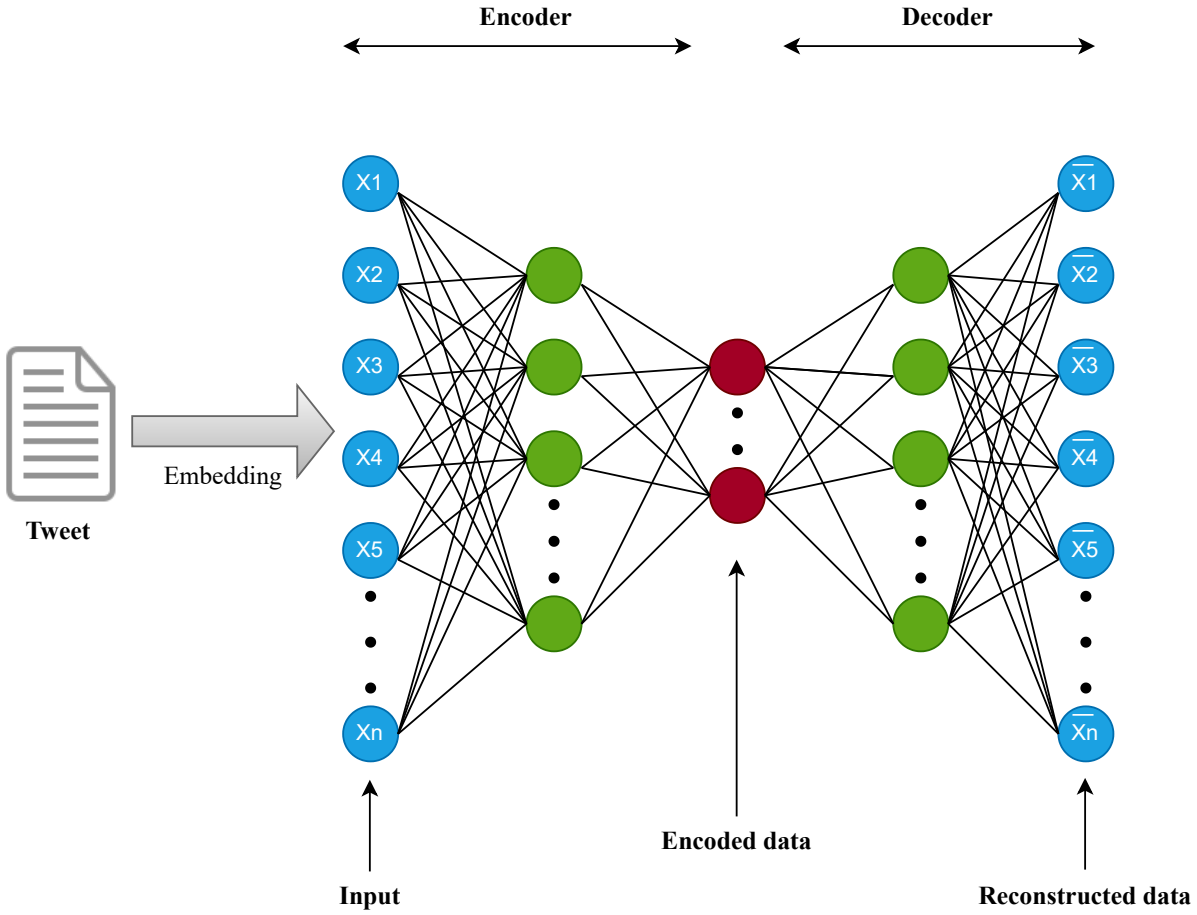

Figure 2.7: Architecture of an Autoencoder.

In our architecture we will only use the encoder part of the autoencoder in order to extract the features from our text. The encoder takes our input text and maps it into a lower dimensional hidden representation that we will further feed to a classification model that we will talk about in the next section to perform the authorship attribution task. The encoder architecture we use is illustrated in figure 2.8.

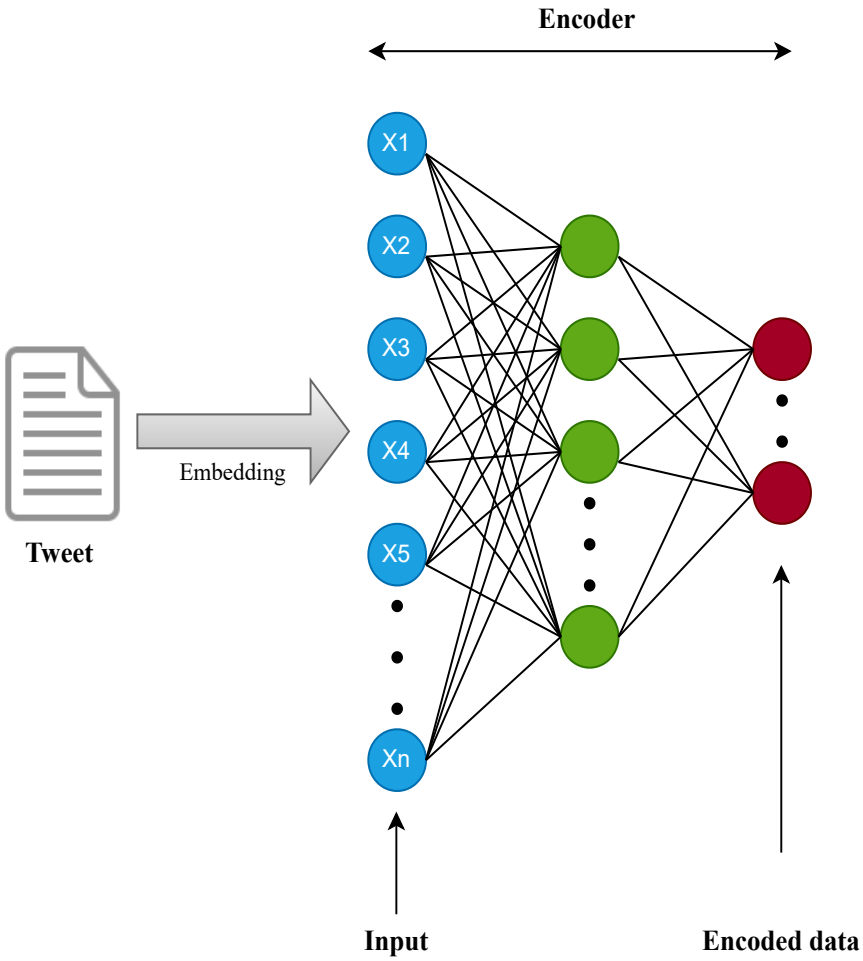

Figure 2.8: The architecture of our encoder.

2.7.7 AdaBoost Classifier

Adaboost short for adaptive boosting is a machine learning and the first practical boosting algorithm, and is now one of the most commonly used and studied, with applications in a wide range of fields [Schapire, 2013].

It is a classification task ensemble method that combines numerous "weak" classifiers to build a strong classifier, the algorithm iteratively trains weak classifiers on different subsets of the training data and throughout each iteration misclassified instances are given greater weights which allows the algorithm to focus on samples that are difficult to classify correctly [Schapire et al., 2000].

The weak classifiers are changed in later iterations to pay greater attention to these difficult

samples, the final classification is established by a weighted combination of the weak classifiers, with the contribution of each classifier weighted based on its performance [Schapire et al., 2000].

In our architecture we use AdaBoost with decision trees as base estimators.

2.7.8 Proposed Autoencoder-Adaboost Architecture

Our third proposed architecture is composed of the combination between the encoder part of an autoencoder for the feature extraction from the input text data, and the second part which is an AdaBoost classifier using decision trees as base estimators for the final authorship attribution classification.

We present an illustration of the architecture in figure 2.9.

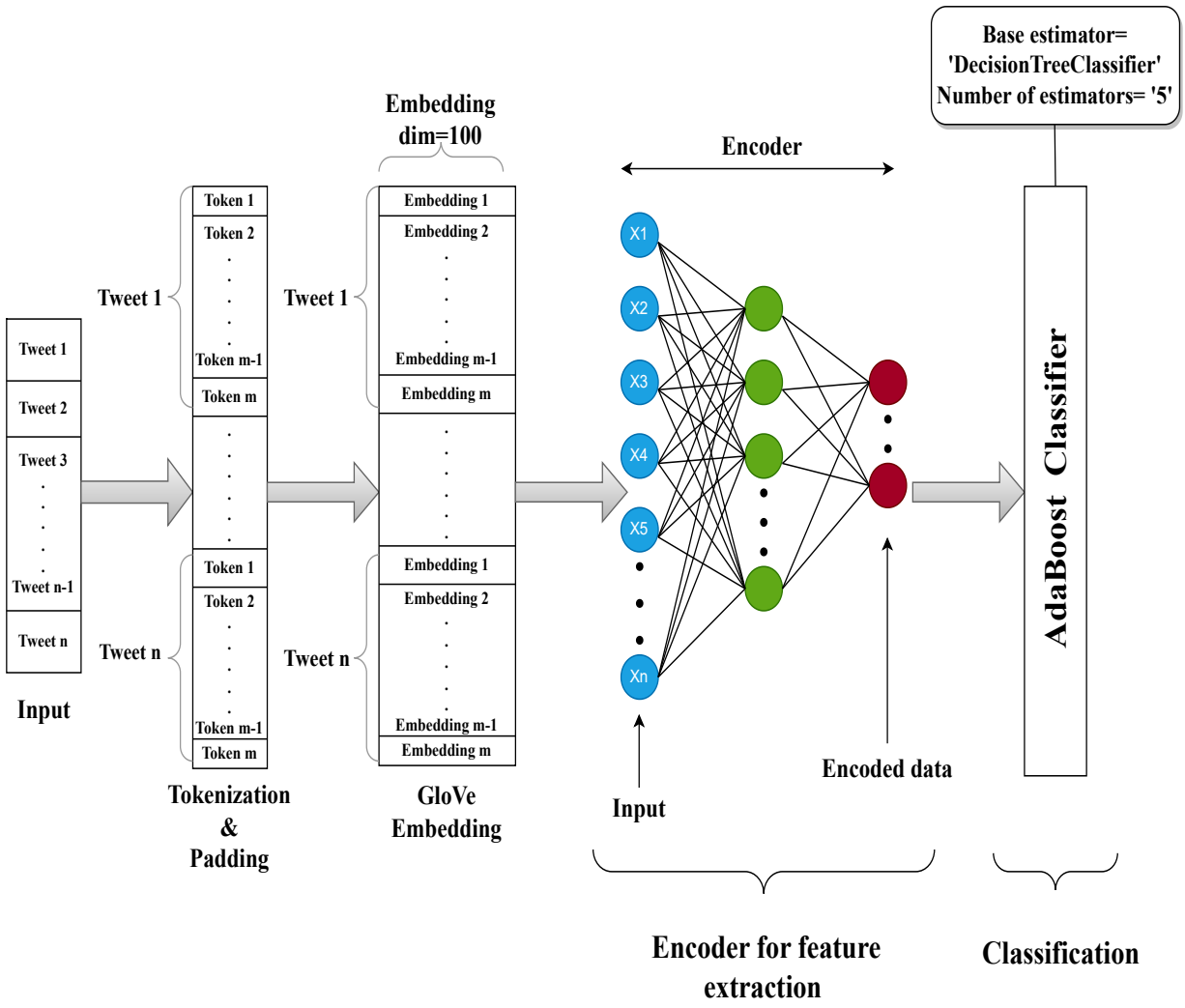

Figure 2.9: Proposed Autoencoder-AdaBoost Architecture for Authorship Attribution.

A detailed description of the architecture is provided in the following part :

- The first part is the input layer which receives the pre-processed and tokenized tweets, the max length is equal to 100.
- The input is then passed to a GloVe embedding layer we use 'glove.twitter.27B' with the embedding dimension equal to 100.
- Next we apply a flatten layer to transform the embedding output to a 1D array.
- A dense layer with 84 nodes and using 'Relu' activation is applied next to map the data to a lower dimensional representation.
- For the final classification we use an AdaBoost classifier with 5 decision trees as base estimators.

2.8 Conclusion

In this chapter we introduced our proposed solution including the different models that we utilized and their respective architectures, we also presented the dataset that we used followed by the pre-processing steps that were taken in order to clean our data. In addition, we explained the process of vectorization, which includes tokenization and word embedding techniques, in detail.

In the next chapter we will present the experimental setup with results achieved by each model, along with the tools and libraries that have been used. We also provide a comparison of the performance of our proposed models to the state of the art results.

Chapter 3

Experimentation And Results

3.1 Introduction

In this chapter we thouroughly examine our proposed models for effectively predicting the authorship of unknown tweets we present their performances and compare them to state of the art results, to accomplish this we used various evaluation metrics that will also be presented. Additionally, we present the experiments conducted and tools and libraries that have been useful during the course of our work.

3.2 Hardware Specifications

In this section we will provide a brief overview of the hardware specifications used in the building and training and evaluation of the machine leaning and deep learning models employed in our work, we used two computers with the following configurations:

For the first PC:

- **Processor:** Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz 1.90 GHz
- **RAM:** 8,00 Go
- **System type:** 64-bit Operating System, x64-based processor

For the second PC:

- **Processor:** Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz

- **RAM:** 8,00 Go
- **System type:** 64-bit Operating System, x64-based processor

3.3 Software Environment

For our experiments we used python 3.10 as a programming language withing the Google Colaboratory environment which we will be talking about in this section along with the various python libraries that have been essential in our work.

3.3.1 Google Colaboratory

Google Colaboratory more commonly known as "colab" [Bisong, 2019], is a research initiative hosted on the Google Cloud platform. It offers a serverless Jupyter notebook environment for interactive development [Bisong, 2019]. Colab offers the ability to prototype machine learning models on powerful hardware such as GPUs and TPUs, thus enabling accelerated computations for data analysis and model training [Bisong, 2019].

3.3.2 Python Libraries

In this section we present the python libraries that played an important role in our work, python being a powerful programming language offers a variety of libraries that helped design and implement our models.

3.3.2.1 NumPy

NumPy [1] is a powerful, free, and open-source library for the Python programming language that provides support for large multi-dimensional arrays, along with a collection of high-level mathematical functions which makes it an essential tool for scientific computing and data analysis.

[1] https://www.nvidia.com/en-us/glossary/data-science/numpy/

3.3.2.2 Pandas

Pandas [2] is the most widely used data manipulation and analysis software open-source library built on the Python programming language, it provides data structures and operations for strong, flexible, and user-friendly data analysis and manipulation. Pandas enhances Python by enabling it to work with data similar to spreadsheets, facilitating quick loading, aligning, manipulating, and merging in addition to other crucial operations.

3.3.2.3 Sickit-learn

Sickit-learn [3] is a powerful and well known machine learning library for python package that includes a large number of algorithms as well as tools for ML visualizations, preprocessing, model fitting, selection, and evaluation. Scikit-learn, which is based on NumPy, SciPy, and matplotlib, includes a number of efficient algorithms for classification, regression, and clustering such as Support vector machines, rain forests, and gradient boosting.

3.3.2.4 Keras

Keras [4] is a robust open-source deep learning library that builds on top of current frameworks like TensorFlow and Theano, it offers a high-level interface for designing and training neural networks, making it easy to build and test different deep learning models.

3.4 Evaluation Metrics

In this section we will delve into the evaluation metrics that we used in order to asses the performance of our model, evaluation metrics serve as a quantitative measure that helps determine the efficacy of our classifiers in predicting the authorship of a given tweet. In our work we focused on four main evaluation metrics namely: accuracy, precision, recall, and F1-score.

Before we define each metric we must first establish a clear understanding of the fundemental elements used in these metrics:

- **True Positive (TP):**

  True positive refers instances that are correctly predicted as positive by the model.

[2]https://www.nvidia.com/en-us/glossary/data-science/pandas-python/
[3]https://www.nvidia.com/en-us/glossary/data-science/scikit-learn/
[4]https://developer.nvidia.com/blog/scaling-keras-training-multiple-gpus/

- **True Negative (TN):**

  It refers to the instances that are correctly predicted as negative by the model.

- **False Positive (FP):**

  This value represents the instances that are negative and were incorrectly predicted as positive by the model.

- **False Negative (FN):**

  This value refers to the instances that despite being positive were falsely predicted as negative by the model.

3.4.1 Accuracy

Accuracy measures the percentage of correctly predicted labels [M and M.N, 2015]. It is defined as follows:

$$\frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

3.4.2 Precision

Precision measures the correctly predicted positive labels from the total predicted labels in a positive class [M and M.N, 2015]. It is defined as follows:

$$\frac{TP}{TP + FP} \tag{3.2}$$

3.4.3 Recall

Recall measures the positive labels that are correctly classified by the model [M and M.N, 2015]. Recall is defined as:

$$\frac{TP}{TP + FN} \tag{3.3}$$

3.4.4 F1-score

The F1 score is a harmonic mean of precision and recall [M and M.N, 2015], it provides a single metric that balances both metrics, its equation is:

$$\frac{2 * Precision * Recall}{Precision + Recall} \tag{3.4}$$

In a multi classification problem such as ours it wouldn't be possible to use precision, recall, and f1-score directly for evaluation, instead we would need to calculate these metrics for each one of the classes individually by treating each class as the positive class and the rest as negative classes. Then, we compute the average of precision, recall, and f1-score across all classes to get the final value for each evaluation metric.

Each of these metrics are available for multi classification problems and are provided by the Sickit-learn library.

3.5 Experimental Setup

In this section we present the experiments and results of our proposed models, the objective is to evaluate the performances of each model and explore the impact of using pure unprocessed tweets compared to pre-processed tweets as well as the influence of using different dataset sizes by considering 50 users and 10 users. At the end of the experiments we provide a comparison between the results achieved by our models and the state of the art results.

3.5.1 Experiment 1: Pure tweets Vs pre-processed tweets

In this experiment we evaluate the three models: TCN, LSTM-CNN, and Autoencoder-Adaboost using a subset of the dataset of 50 users randomly selected. We first test with pure tweets then with pre-processed tweets.

3.5.1.1 Case 1 : Using pure tweets

Prior research has demonstrated the importance of stopwords, URLs, and other symbols [Suman et al., 2022], in order to observe the impact of pre-processing on the data we experiment with first feeding the original tweets directly to the model then with applying pre-processing steps to the tweets.

- **TCN model results:**

  The results achieved by the TCN model are presented in table 3.1, for this model we have used 'Sparse categorical crossentropy' as loss function since we have a multi classification problem, for the optimizer we used 'adam', and a number of epochs equal to '10'.

Table 3.1: TCN performance on pure tweets using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	52.02%	53.29%	52.47%	51.92%

The model achieved 52.02% accuracy and a loss equal to 1.73, figures 3.1 and 3.2 illustrate the accuracy and the loss values throughout the epochs for both training and validation data:

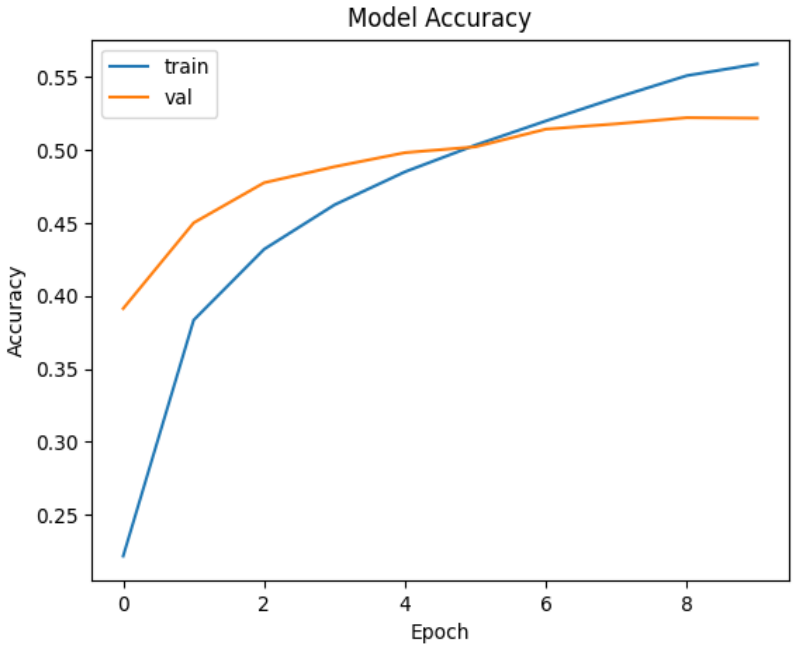

Figure 3.1: Accuracy graph for 10 epochs using TCN and pure tweets.

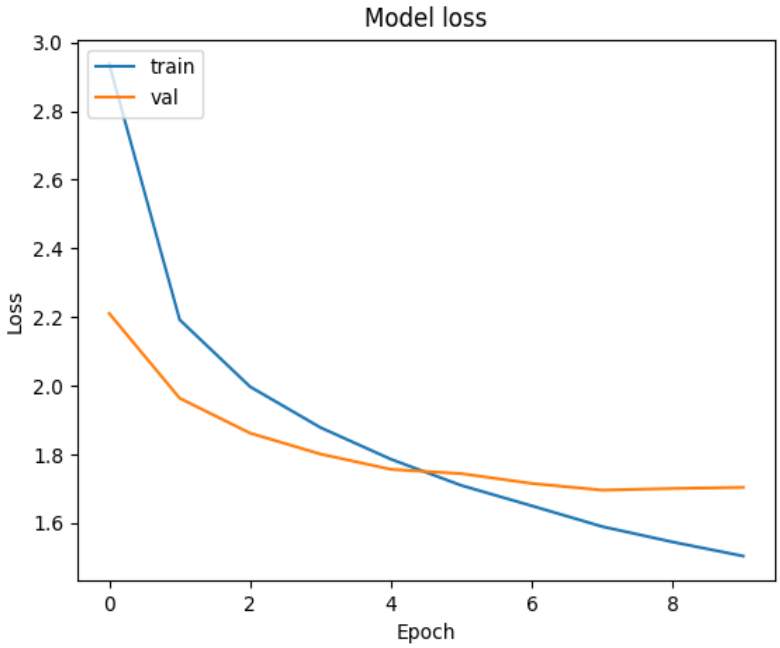

Figure 3.2: Loss graph for 10 epochs using TCN and pure tweets.

- **LSTM-CNN model results:**

  For the LSTM-CNN model we used 'adam' optimizer and 'Sparse categorical crossentropy' as loss function and 10 epochs as well, the results achieved by this model are presented in table 3.2:

Table 3.2: LSTM-CNN performance on pure tweets using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	52.77%	54.26%	52.47%	52.39%

As it can be noticed from the results table, the performances achieved by the TCN and LSTM-CNN model show remarkable resemblance, since both models obtained very similar results across all evaluation metrics.

The accuracy and loss graphs throughout all ephos are presented below in figures 3.3 and 3.4:

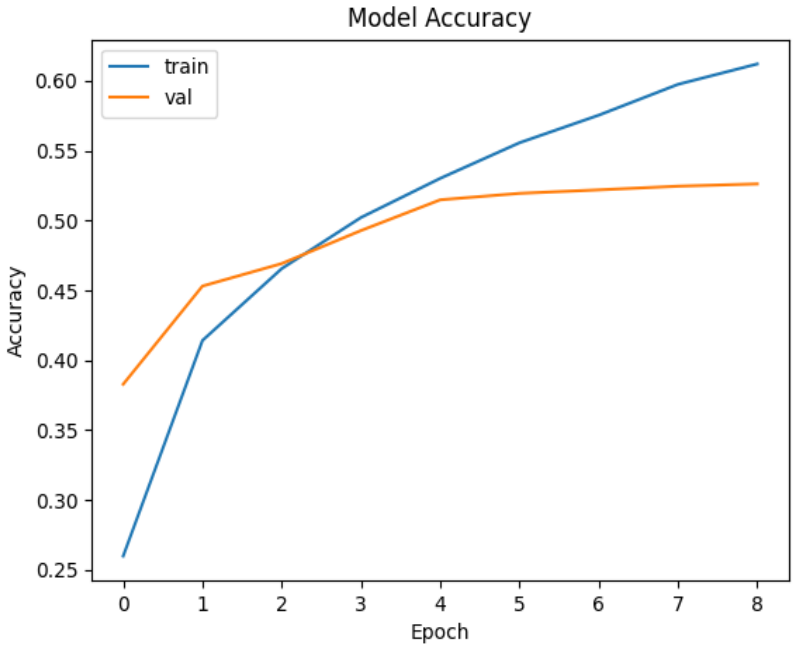

Figure 3.3: Accuracy graph for 10 epochs using LSTM-CNN and pure tweets.

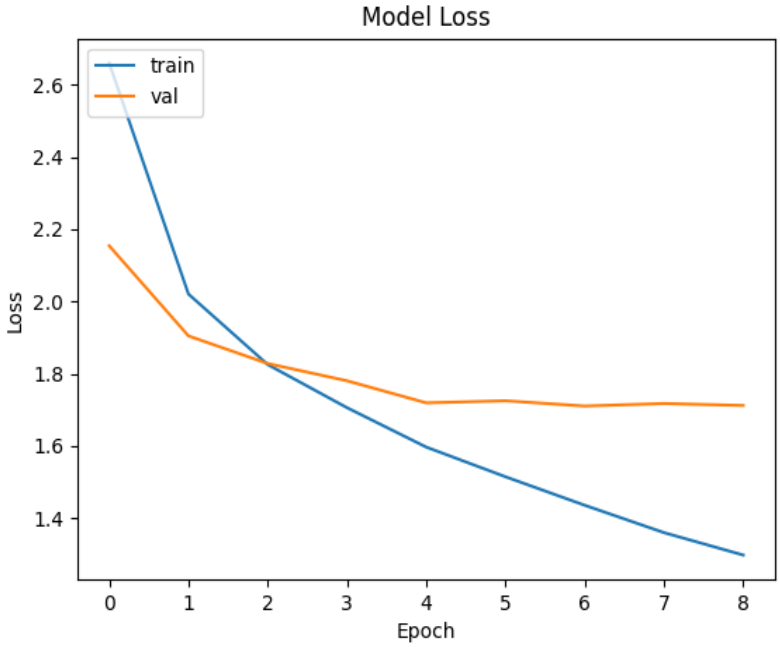

Figure 3.4: Loss graph for 10 epochs using LSTM-CNN and pure tweets.

- **Autoencoder-Adaboost model results:**

  In the encoder part of this model we used 'Mean squared error' for the loss function, and

'adam' optimizer and the number of epochs is 20, we present the results achieved in table 3.3:

Table 3.3: Autoencoder-Adaboost performance on pure tweets using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	17.21%	17.22%	17.19%	16.96%

This model exhibits very poor performances, as shown by the results table the model achieved results considerably worse when compared to the two previous TCN, LSTM-CNN models.

3.5.1.2 Case 2: Using pre-processed tweets

In this experiment, we apply pre-processing steps to our data namely URL removal as well as hashtag, emojis, punctuation and special characters, we present the results achieved by all models using the different metrics previously used.

- **TCN model results:**

  We use the same parameter configuration as the previous experiment in this model in order to effectively asses the impact of the pre-processing on the model's performance, the results after applying the pre-processing steps to our tweets are presented in table 3.4:

Table 3.4: TCN performance on pre-processed tweets using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	43.95%	44.09%	44.03%	43.20%

The model reached an accuracy of 43.95% and a loss value of 2.04, the accuracy and loss graphs for the validation and train data are illustrated in figures 3.5 and 3.6:

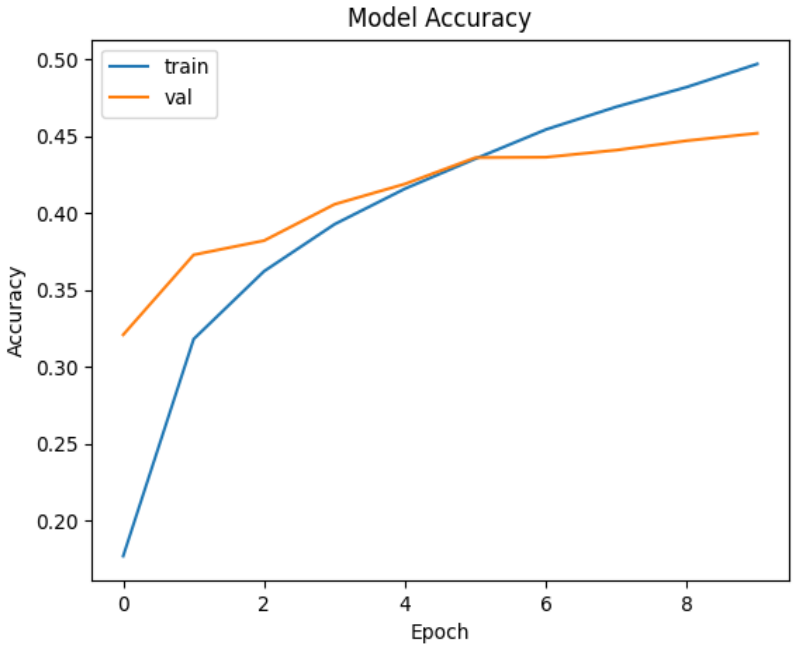

Figure 3.5: Accuracy graph for 10 epochs using TCN and pre-processed tweets.

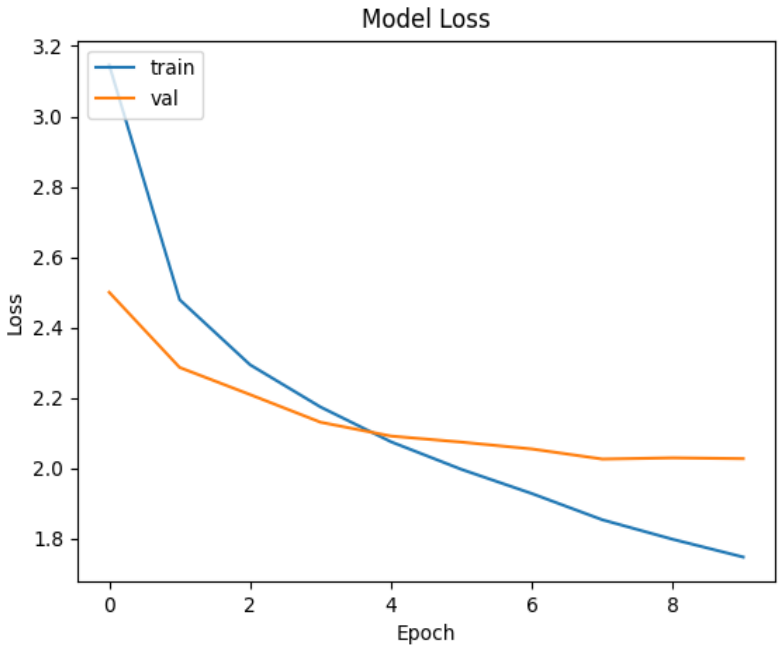

Figure 3.6: Loss graph for 10 epochs using TCN and pre-processed tweets.

- **LSTM-CNN model results:**

For this model we also used the same parameters that have been previously used, we demonstrate the results achieved by this model in table 3.5:

Table 3.5: LSTM-CNN performance on pre-processed tweets using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	42.96%	43.97%	43.05%	42.35%

The accuracy reached by the model was 42.96 and the loss value reached 2.04. The graphs for accuracy and loss with both validation and train data are visualized as follows in figures 3.7 and 3.8:

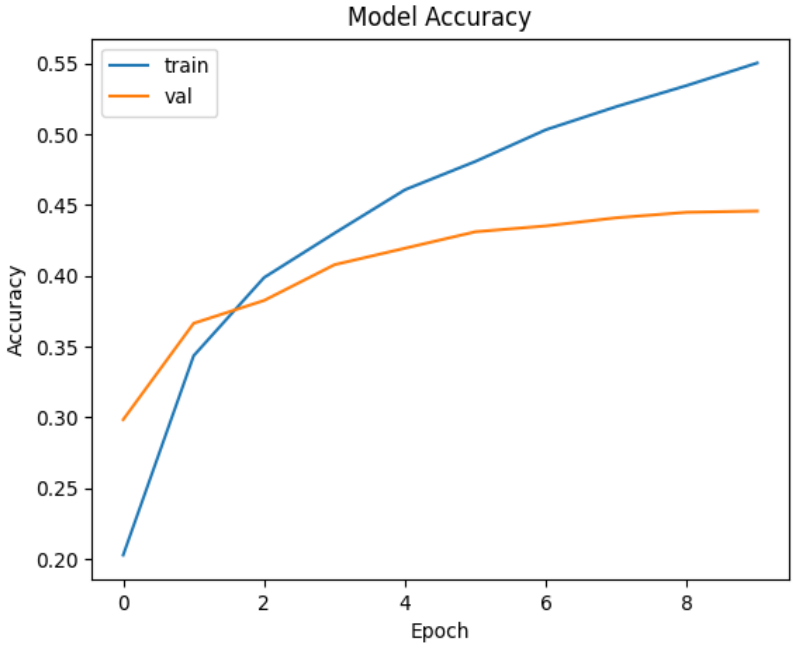

Figure 3.7: Accuracy graph for 10 epochs using LSTM-CNN and pre-processed tweets.

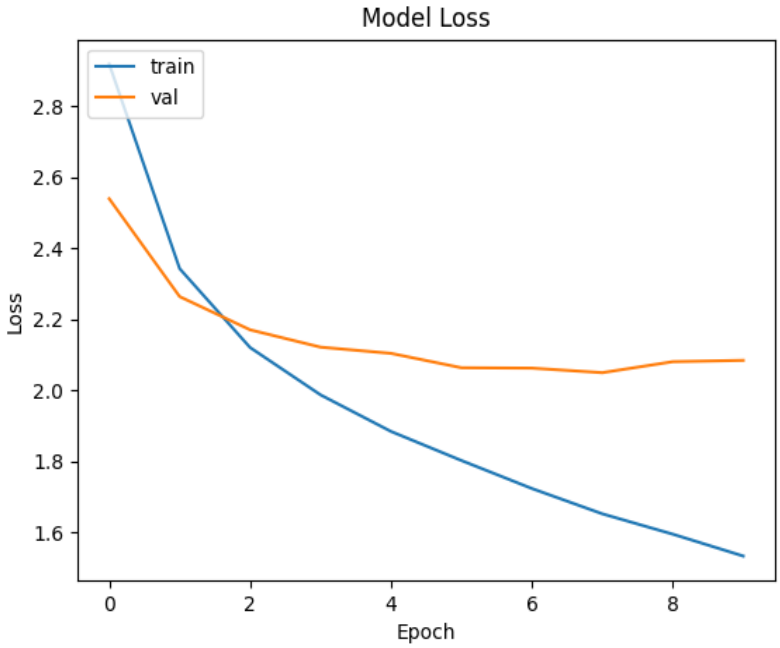

Figure 3.8: Loss graph for 10 epochs using LSTM-CNN and pre-processed tweets.

- **Autoencoder-Adaboost model results:**

  Similarly to the previous two models, we kept the same parameters for this model. We showcase the results achieved using table 3.6:

Table 3.6: Autoencoder-Adaboost performance on pre-processed tweets using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	10.20%	10.01%	10.46%	10.04%

The results of the first experiment indicate that the pre-processing steps employed had a negative impact on the performances of the three models, we were able to observe a decrease in all the evaluation metrics used for the evaluation for the three models when using pre-processed tweets compared to using pure tweets. This observation sheds light on an important aspect in authorship attribution of short texts problems where the normalization steps can potentially take away some important features that could help distinguish individual writing styles and further decrease the performance of the AA models. Table 3.7 represents a comparison between the results achieved using pre-processed tweets and when using pure tweets.

Table 3.7: Comparison between the results of the first experiment

Model	Pure tweets				Pre-processed tweets			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
TCN	52.02%	53.29%	52.47%	51.92%	43.95%	44.09%	44.03%	43.20%
LSLM-CNN	52.77%	54.26%	52.47%	52.39%	42.96%	43.97%	43.05%	42.35%
AE-Adaboost	17.21%	17.22%	17.19%	16.96%	10.20%	10.01%	10.46%	10.04%

Based on these results we will henceforth use unprocessed data for the remaining of the experiments as it helps preserve the features that could be useful in our task.

3.5.2 Experiment 2: Decreasing the number of users

Since previous research has established that the number of authors affects the performance of an authorship attribution model, we conduct this experiment by varying the number of users. in this section we will focus on the case of reducing the number of users to 10, as we have already presented the results that could be achieved using 50 users in the previous experiments.

- **TCN model results:**

  For the TCN model we kept the same parameters as the previous experiments, the results using 10 users as showcased in table 3.7:

  Table 3.8: TCN performance with 10 users using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	81.96%	82.77%	81.61%	82.01%

  The accuracy for this model reached 81.96% while the loss value reached 0.46, below we provide the graphs for accuracy and loss respectively for the training and validaion data in 3.9 and 3.12:

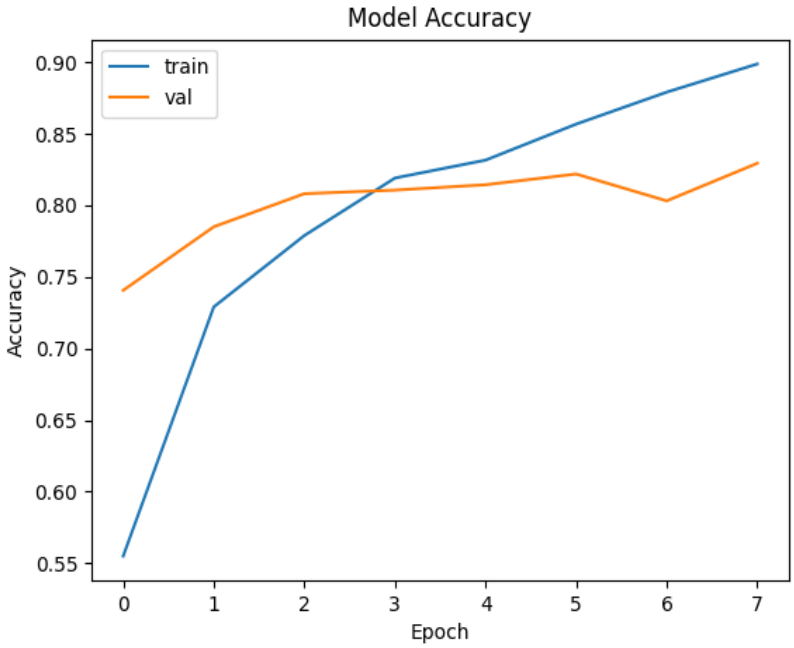

Figure 3.9: Accuracy graph for 10 epochs using TCN with 10 users.

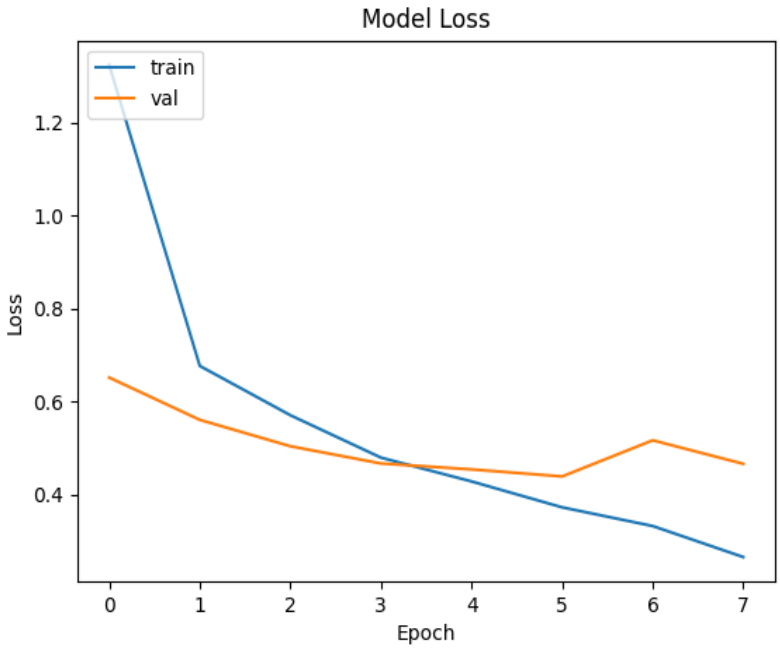

Figure 3.10: Loss graph for 10 epochs using TCN with 10 users.

- **LSTM-CNN model results:**

In this section we present the results of the LSTM-CNN model using 10 users and the same parameters as previous experiments, the results are presented in table 3.8:

Table 3.9: LSTM-CNN performance with 10 users using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	81.95%	81.45%	82.41%	81.57%

The LSTM-CNN model reached an accuracy of 81.95% in this experiment with a loss value of 0.51. Both graphs for accuracy and loss are provided below:

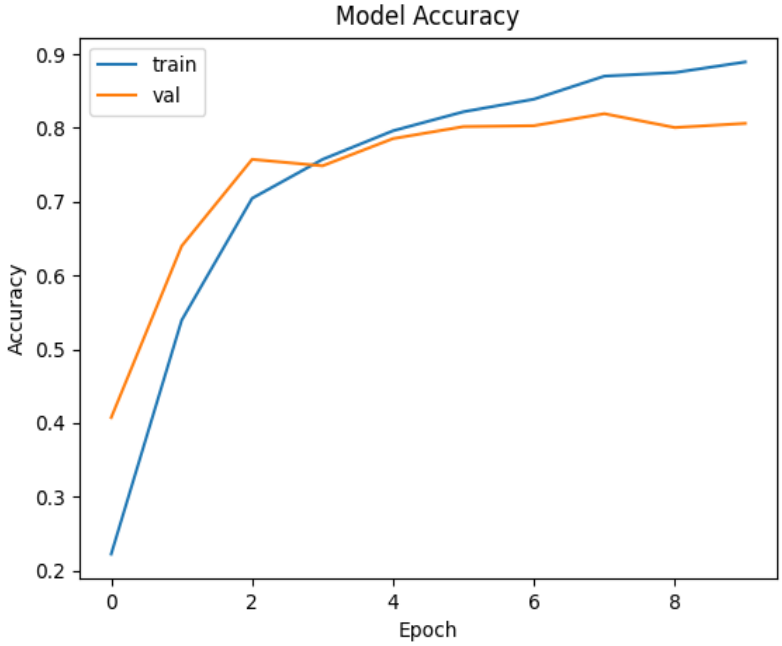

Figure 3.11: Accuracy graph for 10 epochs using LSTM-CNN with 10 users.

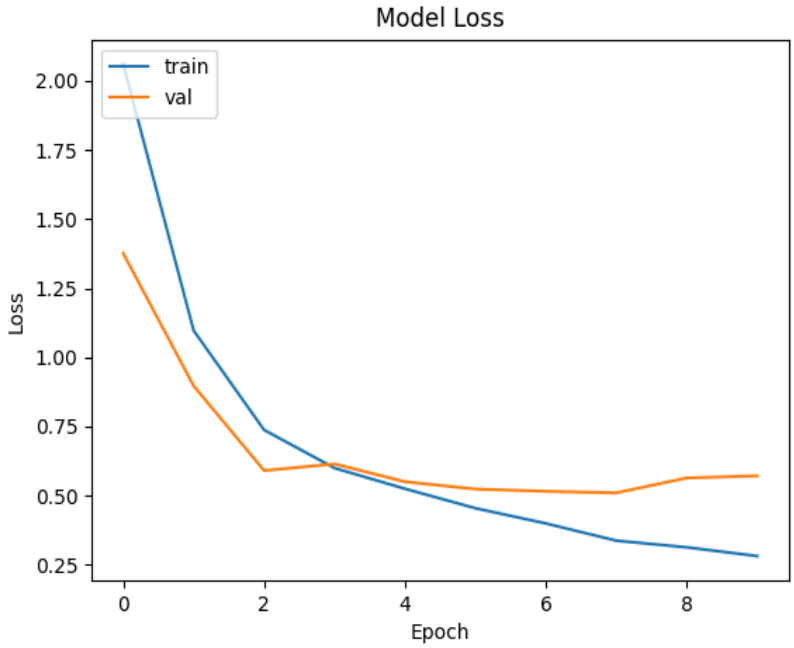

Figure 3.12: Loss graph for 10 epochs using LSTM-CNN with 10 users.

- **Autoencoder-Adaboost model results:**

  In this last part we present the results of the Autoencoder-Adaboost model which has worse results in comparison to the previous two models we evaluated, table 3.9 summarizes its results:

Table 3.10: Autoencoder-Adaboost performance with 10 users using the different evaluation metrics

Evaluation Metric	Accuracy	Precision	Recall	F1-score
Value	46.25%	45.79%	45.76%	45.65%

The experimental results demonstrate a notable improvement in the performance across all three models when the number of users is decreased to 10, this conclusion aligns with the previous research, where it has been proven that decreasing the number of authors enhances the performance of authorship attribution models. Table 3.11 represents a comparison between the results when using 50 users and when using 10 users.

Table 3.11: Comparison between the results of the second experiment

Model	50 Users				10 Users			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
TCN	52.02%	53.29%	52.47%	51.92%	81.96%	82.77%	81.61%	82.01%
LSLM-CNN	52.77%	54.26%	52.47%	52.39%	81.95%	81.45%	82.41%	81.57%
AE-Adaboost	17.21%	17.22%	17.19%	16.96%	46.25%	45.79%	45.76%	45.65%

3.5.3 Comparison With The State Of The Art Results

In this section we provide a comparison of the results achieved across all our three models and the state of the art results, we resume these results in table 3.12 :

Table 3.12: Comparison with the state of the art results

Model	**Accuracy**
Wang and Iwaihara, 2021	88.2%
Huang et al., 2020	83.6%
Suman et al., 2022	86.62%
Shrestha et al., 2017	76.1%
TCN	52.02%
LSTM-CNN	52.77%
Autoencoder-Adaboost	17.21%

Despite our efforts our models fell short of achieving state of the art results, the obtained results for the three models indicate that there is still room for improvement and also highlight the complexity of the authorship attribution task. While both the TCN and LSTM-CNN models had similar and moderate results (52.02% and 52.77% accuracy), the Autoencoder-Adaboost model performed significantly worse in comparison (17.21% accuracy) suggesting that it struggled in capturing necessary features and unique writing styles.

3.6 Application Interface

In this section we present our authorship attribution application's interface, first figure 3.13 illustrates the interface of the application at launch:

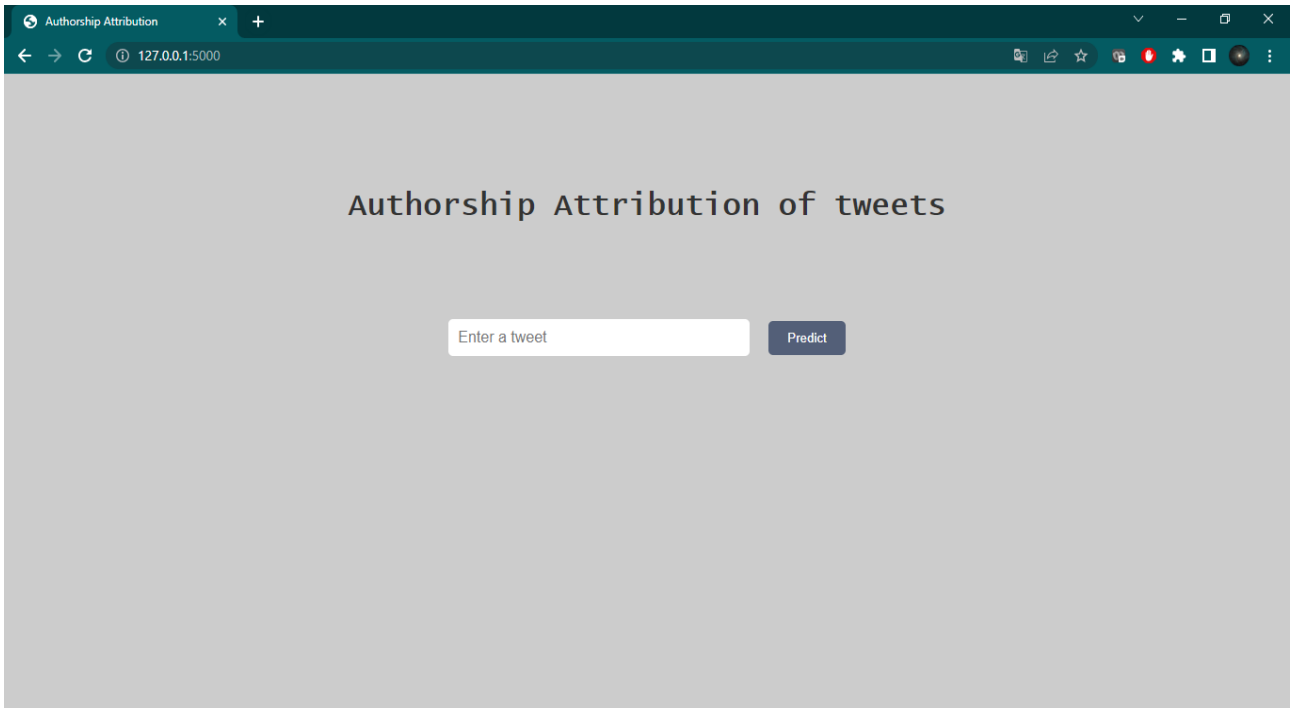

Figure 3.13: The interface at the launch of the application.

From here the user can enter a tweet in the input field and in order to predict its authorship the user must click on the button labeled "predict" and the output will be displayed below. We provide an example of a tweet's author prediction in the following figure:

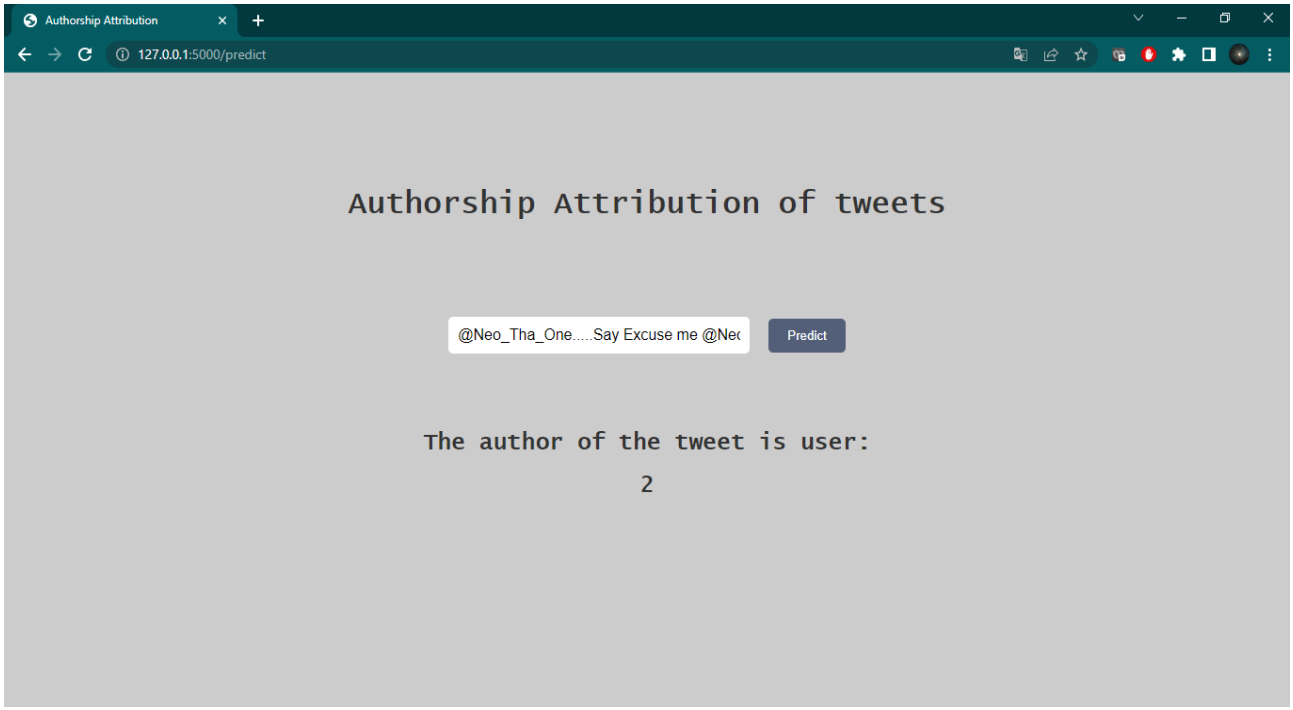

Figure 3.14: The interface using an example of the authorship attribution of a tweet.

3.7 Conclusion

In conclusion, in this chapter, we presented the various libraries that have been used in our models, as well as the evaluation metrics that we employed to asses their performances. Next we presented in detail the experiments that we conducted while also providing the results achieved across all three models. Additionally, we provided a comparison between our proposed models and the state of the art results and also shed light on the complexity of the authorship attribution task as seen by the results achieved.

In the end we presented our user interface with an example of the execution and the authorship attribution of an anonymous tweet.

General Conclusion

With the exponential growth of social media usage around the world a multitude of issues and crimes emerged, including online harassment, fake news spread, identity theft, and a numerous other pressing issues. Addressing these problems has become crucial in order to provide a safe environment for social media users assuring trustworthiness and accountability.

Authorship attribution on social networks proves to be an effective solution to these problems by accurately attributing authorship to the content published on social networks, we can hold users accountable for their actions in these platforms.

The purpose of our work is to propose an authorship attribution model that can effectively predict the author of unknown post of social network platforms, using machine learning and deep learning algorithms. We proposed three distinct models to tackle this problem, namely TCN which we were the first to use in the task of authorship attribution, LSTM combined with CNN, and Autoencoder combined with Adaboost. Our models were evaluated using data from a Twitter dataset containing over 7 million tweets from which we selected and experimented with a varying number of tweets and users, we conducted several experiments in order to asses the performances of our models across different scenarios and settings.

Despite utilizing advanced machine learning and deep learning algorithms, we faced difficulties in achieving satisfactory results with the TCN model reaching 52.02% accuracy and the LSTM-CNN reaching a similar value of 52.77%, the worst performing model was the Autoencoder combined with Adaboost which reached an accuracy of only 17.21%. These low performances can be the consequence of the complexity of the authorship attribution problem applied to social networks due to the variation of users' writing styles and the briefness of the content present on Twitter as it imposes a limit of 280 characters per tweet, Moreover, our experiments have proven that the task of authorship attribution becomes significantly more challenging as the number of authors increases, we observed how the results varied when altering the number of users. As the number of authors grew, the complexity of distinguishing individual writing styles and identifying distinct patterns amplified.

Our future perspectives in this field include improving the accuracy by considering other approaches that may produce better performances, we would also like to experiment on larger datasets beyond Twitter such as other social media platforms. Another interesting research direction would be to extend our approach to other languages namely the Arabic language which is known for its rich and unique characteristics. By pursuing these directions we aim to improve and make significant contributions in the authorship attribution field.

Bibliography

[Abbasi and Chen, 2005] Abbasi, A. and Chen, H. (2005). Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20(5):67–75.

[Arras et al., 2019] Arras, L., Arjona-Medina, J., Widrich, M., Montavon, G., Gillhofer, M., Muller, K.-R., Hochreiter, S., and Samek, W. (2019). *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700. Springer International Publishing.

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

[Bird et al., 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media , Inc.

[Birunda and Devi, 2021] Birunda, S. S. and Devi, R. K. (2021). *A review on word embedding techniques for text classification*, volume 59, pages 267–281. Springer, Singapore.

[Bisong, 2019] Bisong, E. (2019). *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Apress.

[Chowdhury, 2003] Chowdhury, G. G. (2003). Natural language processing.

[Coughlin, 1990] Coughlin, J. (1990). Perspectives on natural language processing. *The French Review*, 64(1):172–179.

[Dean, 2023] Dean, B. (2023). Social network usage growth statistics: How many people use social media in 2023? Available online. Accessed on June 17, 2023.

[Deng and Liu, 2018] Deng, L. and Liu, Y. (2018). *Deep Learning in Natural Language Processing*. Springer Singapore.

[Eder et al., 2016] Eder, M., Rybicki, J., and Kestemont, M. (2016). Stylometry with r: A package for computational text analysis.

[ELMANARELBOUANANI and KASSOU, 2014] ELMANARELBOUANANI, S. and KASSOU, I. (2014). Authorship analysis studies: A survey. *International Journal of Computer Applications*, 86:22–29.

[Fabien et al., 2020] Fabien, M., Villatoro-Tello, E., Motlicek, P., and Parida, S. (2020). Bertaa: Bert fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137. NLP Association of India (NLPAI).

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Huang et al., 2020] Huang, W., Su, R., and Iwaihara, M. (2020). Contribution of improved character embedding and latent posting styles to authorship attribution of short texts. volume 12318 LNCS, pages 261–269. Springer Science and Business Media Deutschland GmbH.

[Juola, 2008] Juola, P. (2008). Authorship attribution. *Foundations and Trends in Information Retrieval*, 1(3):233–334.

[Kah et al., 2022] Kah, A. E., Airej, A. E., and Zeroual, I. (2022). Arabic authorship attribution on twitter: what is really matters? *Indonesian Journal of Electrical Engineering and Computer Science*, 28:1730–1737.

[Kaur et al., 2019] Kaur, R., Singh, S., and Kumar, H. (2019). *Authorship Analysis of Online Social Media Content*, volume 46, pages 539–549. Springer.

[Lam et al., 2021] Lam, T., Demange, J., and Longhi, J. (2021). Attribution d'auteur par utilisation des méthodes d'apprentissage profond. In *Proceedings of EGC 2021 Atelier "DL for NLP: Deep Learning pour le traitement automatique des langues"*, Montpellier, France.

[Lara-Benítez et al., 2020] Lara-Benítez, P., Carranza-García, M., Luna-Romera, J. M., and Riquelme, J. C. (2020). Temporal convolutional networks applied to energy-related time series forecasting. *Applied Sciences (Switzerland)*, 10.

[Li et al., 2017] Li, X., Peng, L., Yao, X., Cui, S., Hu, Y., You, C., and Chi, T. (2017). Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation. *Environmental Pollution*, 231:997–1004.

[M and M.N, 2015] M, H. and M.N, S. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining Knowledge Management Process*, 5:01–11.

[Mendenhall, 1887] Mendenhall, T. C. (1887). The characteristic curves of composition. *Science*, ns-9(214s):237–246.

[Mosteller and Wallace, 1963] Mosteller, F. and Wallace, D. L. (1963). Inference in an authorship problem. *Journal of the American Statistical Association*, 58(302):275–309.

[Muttenthaler et al., 2019] Muttenthaler, L., Lucas, G., and janek Amann (2019). Authorship attribution in fan-fictional texts given variable length character and word n-grams notebook for pan at clef 2019.

[Naili et al., 2017] Naili, M., Chaibi, A. H., and Ghezala, H. H. B. (2017). Comparative study of word embedding methods in topic segmentation. volume 112, pages 340–349. Elsevier B.V.

[Onan, 2022] Onan, A. (2022). Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification. *Journal of King Saud University - Computer and Information Sciences*, 34:2098–2117.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

[Petrosyan, 2023] Petrosyan, A. (2023). Worldwide digital population 2023. Available online. Accessed on June 17, 2023.

[Rabab'ah et al., 2016] Rabab'ah, A., Al-Ayyoub, M., Jararweh, Y., and Aldwairi, M. (2016). Authorship attribution of arabic tweets. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–6.

[Rangel et al., 2013] Rangel, F., Rosso, P., Koppel, M., Stamatatos, E., and Inches, G. (2013). Overview of the author profiling task at pan 2013.

[Salvaris et al., 2018] Salvaris, M., Dean, D., and Tok, W. H. (2018). *Deep learning with azure: Building and deploying artificial intelligence solutions on the microsoft AI platform*. Apress Media LLC.

[Sari et al., 2017] Sari, Y., Vlachos, A., and Stevenson, M. (2017). Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 267–273. Association for Computational Linguistics.

[Schapire, 2013] Schapire, R. E. (2013). Explaining adaboost. In *Empirical Inference*.

[Schapire et al., 2000] Schapire, R. E., Carbonell, J., and Yang, Y. (2000). Boostexter: A boosting-based system for text categorization. 39:135–168.

[Schwartz et al., 2013] Schwartz, R., Tsur, O., Rappoport, A., and Koppel, M. (2013). Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891. Association for Computational Linguistics.

[Sharp et al., 2017] Sharp, B., Sèdes, F., and Lubaszewski, W. (2017). *Cognitive Approach to Natural Language Processing*.

[Shrestha et al., 2017] Shrestha, P., Sierra, S., González, F. A., Rosso, P., Montes-Y-Gómez, M., and Solorio, T. (2017). Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 669–674.

[Stamatatos, 2009] Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60:538–556.

[Stamatatos, 2016] Stamatatos, E. (2016). Authorship verification: A review of recent advances. *Research in Computing Science*, 123:9–25.

[Suman et al., 2022] Suman, C., Raj, A., Saha, S., and Bhattacharyya, P. (2022). Authorship attribution of microtext using capsule networks. *IEEE Transactions on Computational Social Systems*, 9:1038–1047.

[Theophilo et al., 2022] Theophilo, A., Padilha, R., Andaló, F. A., and Rocha, A. (2022). Explainable artificial intelligence for authorship attribution on social media. volume 2022-May, pages 2909–2913. Institute of Electrical and Electronics Engineers Inc.

[Wang and Iwaihara, 2021] Wang, X. and Iwaihara, M. (2021). Integrating roberta fine-tuning and user writing styles for authorship attribution of short texts. volume 12858 LNCS, pages 413–421. Springer Science and Business Media Deutschland GmbH.

[Watson, 2023] Watson, A. (2023). Most trusted sources of news and info worldwide 2022. Available online.

[Yamashita et al., 2018] Yamashita, R., Nishio, M., Do, R. K. G., and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9:611–629.

[Yule, 1939] Yule, G. U. (1939). On sentence- length as a statistical characteristic of style in prose: With application to two cases of disputed authorship. *Biometrika*, 30:363–390.

[Zhang et al., 2018] Zhang, H., Nie, P., Wen, Y., and Yuan, X. (2018). Authorship attribution for short texts with author-document topic model. volume 11061 LNAI, pages 29–41. Springer Verlag.

[Zhu et al., 2022] Zhu, J., Su, L., and Li, Y. (2022). Wind power forecasting based on new hybrid model with tcn residual modification. *Energy and AI*, 10.

[Zipf, 1932] Zipf, G. K. (1932). *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press, Cambridge, MA and London, England.