

Software Engineering Department

Ort Braude College

Course 61998: Extended Project in Software Engineering

**Pre-Trained Authorship Representation Transformer**

In Partial Fulfillment of the Requirements for

Final Project in Software Engineering (Course 61998)

Karmiel – January 2023

Meitar El-Ezra 304680218

Tuval Zitelbach 316219633

**Supervisor**:

Mr. Zeev Volkovich

Mrs. Renata Avros

Contents

[1. ABSTRACT 3](#_Toc121275201)

[2. INTRODUCTION 4](#_Toc121275202)

[3. ARCHITECTURE 5](#_Toc121275203)

[3.1 PRE PROCESSING 5](#_Toc121275204)

[3.2 WORD EMBEDDINGS 6](#_Toc121275205)

# 1. ABSTRACT

Authorship attribution (AA) is the computational task of identifying the author of a text based on a set of possible candidates. Authors imprint intentional and unintentional traces in the form of linguistic features such as punctuation, registry and semantics which can be used to profile authorship. Previous works attempted to encode the semantic features. Those approaches led to poor results on open-set authors (authors that were not included in the training phase). The following approach, nonetheless, attempts to encode features from both context and semantics, focusing more on style rather than semantics alone, which creates what is called, in this article, **authorship embeddings**. PART: contrastively Pre-Trained Authorship Transformer uses zero shot generalization capabilities in authorship identification to compute authorship embeddings with the assistance of the state-of-the-art contextual based models. Analyzing those attribution can contribute to several fields such as forensics, social networks analysis, identifying fake news or profiles and more.

**Keywords:** Authorship attribution, Neural networks, Transformers, Contrastive pretraining.

# 2. INTRODUCTION

Authorship identification of handwritten documents started approximately in the late 19th century; however most textual data is digital. Typically, the input for AA models is a set of candidates and number of texts for each candidate. The objective is to assign text to one of the candidates. AA is divided to three forms. First, closed-set attribution, where the true author of a given text must be included in the set of suspects. Second, open-set attribution, where the true author of a given text could be excluded from the set of candidates. The third approach, the author verification where there is only one candidate author.

In real life scenarios, the training set of texts and the test set of texts may be differed in the topic and genre. Those situations are examined in the Cross-Domain AA. Most of the previous works in AA focus on the closed-set AA. Nevertheless, the authorship embedding, that is presented in this paper, is calculated with zero-shot generalization capabilities in authorship identification, namely, open-set AA form. In this case, the model gets test set of texts, from both new and known authors, topics, genres, etc.

# 3. ARCHITECTURE

## 3.1 PRE PROCESSING

Before using the data for analysis or prediction, processing the data is important. In order to prepare the text data for the model building we perform text preprocessing. Some of the preprocessing steps are:

* Removing punctuations like (@, !, $, \*, %, . etc.)
* Removing URLs
* Removing Stop words
* Lower casing
* Tokenization
* Stemming
* Lemmatization

In PART architecture design, the data is pre-tokenized using the tokenizer from the NLTK library. This tokenizer is based on several tokenization methods such as: White Space Tokenization, Dictionary based Tokenization, Ruled-Based Tokenization, Regular Expression Tokenization, Penn Treebank Tokenization, Spacy Tokenization, Moses Tokenization and Subword Tokenization.

The databases used for PART are :

* Standardized Gutenberg **(authored books)**
* Blog authorship **(blog posts)**
* Enron mails **(authored corporate Emails)**

**3.1.1 Chunking**

Each author documents-set is merged with a separator token in-between, resulting in a large pre-tokenized text written in its entirety by this author. The pre-tokenized data is being split in chunks of L=512 tokens. If an author ends up with 2 chunks or more to his name, then each chunk is considered separately for training. Otherwise, the author is being excluded.  
  
As part of the pre-processing of the data, the first and last chunk of every book from the Standardized Gutenberg database is being dropped out, because of the identifying information about the author and the book they usually contain. As well as the Email headers and footers from the Emails on the Enron mails database.

**3.1.2 Tokenization**

The tokenizer used in this model was the RoBERTa – large's tokenization algorithm – Byte Pair Encoding (BPE). The BPE is a popular algorithm for subword tokenization. The main goal of this algorithm is to find a way to represent your entire text dataset with the least amount of tokens.   
After the tokenization the 512 tokens are transferred to the next level in the architecture.

## 3.2 WORD EMBEDDINGS

BERT’S architecture is based on part of the Transformer architecture – the encoder. BERT model is built by stacking up Transformer’s encoders. BERT is pretrained on 2 tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

The model uses a frozen transformer. The freezing of the transformer allows it to keep interpreting language as trained by the masked Language Modeling loss.  
From the tokens it's given, the frozen transformer produces what is called semantic Word Embeddings for the training of the model, with speed. The transformer can maintain all it's original abilities.

The chunk of L=512 tokens are sent to a frozen pre-trained transformer which outputs a matrix in the dimension of LxK, containing the L semantic word embeddings sized K; a word embedding of K features for each input token.

## 3.3 BiLSTM

Bidirectional LSTM (BiLSTM) is a recurrent neural network used in NLP. Unlike standard LSTM, the input flows in both directions, and it’s capable of utilizing information from both sides. It’s also a powerful tool for modeling the sequential dependencies between words and phrases in both directions of the sequence.

In summary, BiLSTM adds one more LSTM layer, which reverses the direction of information flow. Briefly, it means that the input sequence flows backward in the additional LSTM layer. Then we combine the outputs from both LSTM layers in several ways, such as average, sum, multiplication, or concatenation.  
every component of an input sequence has information from both the past and present. For this reason, BiLSTM can produce a more meaningful output, combining LSTM layers from both directions.

the semantic word embeddings have to be interpreted, and for this purpose we append a bidirectional LSTM to the architecture. The BiLSTM is more efficient on lower amounts data points that a transformer layer, therefore the end representations are obtained with a recurrent network. We extract K/2 features for each LSTM pass, to form an embedding of size K.

In order to interpret the L word embeddings, they are sent as L sequences to a BiLSTM, where each K sized word embedding passes through the forward LSTM pass and through the backward LSTM. The passes output 2 features vectors sized K/2, which we then concatenate into one K sized final Embedding.