**DOKUZ EYLÜL UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# AUTHOR IDENTIFICATION WITH TRANSFER LEARNING

**by**

**İbrahim YÜLÜCE**

**June, 2023**

**İZMİR**

# AUTHOR IDENTIFICATION WITH TRANSFER LEARNING

**A Thesis Submitted to the**
**Graduate School of Natural and Applied Sciences of Dokuz Eylül University**
**In Partial FulFillment of the Requirements for the Master of Science in**
**Computer Engineering**

**by**
**İbrahim YÜLÜCE**

**June, 2023**
**İZMİR**

**M.Sc THESIS EXAMINATION RESULT FROM**

We have read the thesis entitled **"AUTHOR IDENTIFICATION WITH TRANSFER LEARNING"** completed by **İBRAHİM YÜLÜCE** under supervision of **ASST. PROF. DR. FERİŞTAH DALKILIÇ** and we certify taht in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Feriştah DALKILIÇ

Supervisor

Assoc. Prof. Omer AYDIN

Thesis Committee Member

Prof. Dr. Alp KUT

Thesis Committee Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

## ACKNOWLEDGEMENTS

First of all, I would like to thank my academic advisor, Asst. Prof. Dr. Feriştah Dalkılıç for her support and continuous guidance throughout my research. She always show me right direction on my research. During my research, she helped me, how the dataset should be prepared, supporting my attandance at conference and preparing high quality product.

I would also like to thank my parents and my sister who have always supported and believed in me. Without them, I would not be able to reach the success I have now.

İbrahim YÜLÜCE

## AUTHOR IDENTIFICATION WITH TRANSFER LEARNING


## ABSTRACT

Author identification is one of the application areas of text mining. It deals with the automatic prediction of the potential author of an electronic text among predefined author candidates by using author specific writing styles. In this study, we conducted an experiment for the identification of the author of a Turkish language text by using transfer learning methods and also machine learning algorithms are used to compare the performance of transfer learning methods. We used GPT-2, BERT and some of classical machine learning algorithms such as Support Vector Machines, Gaussian Naive Bayes, Multi Layer Perceptron, Logistic Regression, Stochastic Gradient Descent and ensemble learning methods including Extremely Randomized Trees and eXtreme Gradient Boosting.

The proposed method was applied on three different sizes of author groups including 10, 15 and 20 authors obtained from a new dataset of newspaper articles. Term frequency-inverse document frequency vectors were created by using 1-gram and 2-gram word tokens. Our results show that the most successful method is the BERT with a classification performance accuracy of 98.6% in 10 authors. SGD is the most successful method in 15 and 20 authors with 97.5% and 97.1% accuracy performance.



**Keywords:** Author identification, transfer learning, machine learning, natural language processing, text mining

# TRANSFER ÖĞRENİMİ İLE YAZAR TANIMA

## ÖZ

Yazar tanıma, metin madenciliğinin uygulama alanlarından biridir. Bir elektronik metnin potansiyel yazarının önceden tanımlanmış yazar adayları arasından yazara özgü yazı stilleri kullanılarak otomatik olarak tahmin edilmesi ile ilgilenir. Bu çalışmada, Türkçe bir metnin yazarının belirlenmesi için transfer öğrenme yöntemleri ve ayrıca transfer öğrenme yöntemlerinin performansını karşılaştırmak için makine öğrenmesi algoritmaları kullanılmıştır. GPT-2, BERT ve Support Vector Machines, Gaussian Naive Bayes, Multi Layer Perceptron, Logistic Regression, Stokastik Gradient Descent gibi klasik makine öğrenimi algoritmalarından bazılarını ve Extremely Randomized Trees ve eXtreme Gradient Boosting dahil topluluk öğrenme yöntemlerini kullandık.

Önerilen yöntem, yeni bir gazete makalesi veri setinden elde edilen 10, 15 ve 20 yazar olmak üzere üç farklı büyüklükteki yazar grubuna uygulanmıştır. 1-gram ve 2-gram kelime belirteçleri kullanılarak terim frekansı-ters belge frekans vektörleri oluşturulmuştur. Sonuçlarımız, en başarılı yöntemin 10 yazarda %98,6 sınıflandırma performans doğruluğu ile BERT olduğunu göstermektedir. SGD %97,5 ve %97,1 doğruluk performansı ile 15 ve 20 yazar arasında en başarılı yöntemdir.

**Anahtar kelimeler:** Yazar tanıma, transfer öğrenimi, makine öğrenmesi, doğal dil işleme, veri madenciliği

**CONTENTS**

**Page**

## LIST OF FIGURES

**Page**

## LIST OF TABLES

**Page**

## ABBREVIATIONS

| | |
|---|---|
| BERT | :Bidirectional Encoder Representations from Transformers |
| GPT | :Generative Pretrained Transformer |
| SVM | :Support Vector Machine |
| RCV1 | :Reuters Corpus Volume I |
| LR | :Linear Regression |
| ELMO | :Embeddings from Language Model |
| TF-IDF | :Term Frequency–Inverse Document Frequency |
| NLP | :Natural Language Processing |
| CBOW | :Continuous Bag of Words |
| IMDB | :Internet Movie Database |
| DNN | :Deep Neural Network |
| ULMFIT | :Universal Language Model Fine-tuning |
| URL | :Uniform Resource Locator |
| HTML | :Hypertext Markup Language |
| TF | :Term Frequency |
| IDF | :Inverse Document Frequency |
| NB | :Naive Bayes |
| XOR | :Exclusive Or |
| API | :Application Programming Interface |
| SGD | :Stochastic Gradient Descent |
| NLTK | :Natural Language Tool Kit |
| MB | :Mega Byte |

# CHAPTER ONE
# INTRODUCTION

## 1.1 Motivation

Author profiling, authorship verification, and author identification are the applications of text mining. Author profiling is the examination of authors' texts to determine their class, including gender, age group etc. Authorship verification is the task of determining whether two or more texts were written by the same author by analyzing linguistic patterns. Author identification deals with estimating the author of an anonymous text from a predefined set of candidate authors.

In this study, we deal with the authorship identification task that is used in many areas including literary studies, history and forensic linguistics. The need to identify the content creator on the internet, detect plagiarism and prevent copyright infringement has increased the interest in authorship identification. In the identification process, stylometric features expose the patterns that appear in the texts belonging to a specific author. Various number of features have been presented including vocabulary richness measures, syntactical features, function words frequencies, character n-gram frequencies, latent semantic analysis (LSA), and Bag-of-Words (BOW) (Stamatatos, 2009; Alhuqail, 2021; Maël, Esau, Petr, & Shantipriya, 2020) in many previous studies. In addition, deep learning and machine learning based methods have been employed for the feature extraction and author identification tasks in recent studies (Mohsen, Nagwa & Nagia 2016).

There are many studies in this field and on this topic. however, among these studies, the number of studies that applied transfer learning methods on their own datasets was small. Since most researchers work with ready-made datasets, we observed that accuracy scores are similar depending on this.

We would both download our own data and prepare a dataset of our own, and we would have the chance to make comparisons by using a method such as transfer

learning, which has been used quite a lot recently. This has always kept our motivation at a high level.

### 1.2 Problem Definition

Author identification has become popular recently, however as technology becomes more common in our lives, it is a method that can be used to create many fake profiles on Twitter or different social media sites and to identify these people. Because even if there is a fake profile, a person analysis can be made from the way the person conveys an emotion or thought. and improvements have been made by researchers using various machine learning algorithms and deep learning methods.

In this study, we deal with the authorship identification task that is used in many areas including literary studies, history and forensic linguistics. The need to identify the content creator on the internet, detect plagiarism and prevent copyright infringement has increased the interest in authorship identification. In the identification process, stylometric features expose the patterns that appear in the texts belonging to a specific author.

It also appears in forensic cases. Whether a found physical evidence or text belongs to that person can be understood both from the handwriting and from the writing style of the person. It can be understood whether the suspect's previously written texts are given to the algorithm and whether it belongs to that person. This event is easier to do in e-texts, texts in the computer environment.

Many researchers have worked on ready-made datasets, these ready-made datasets have examples in many languages. To give an example, there are 50 authors and 2500 articles in the Reuters50 dataset. Since similar methods and algorithms were used by the researchers who carried out studies prepared with these ready-made datasets, there was not much difference as a result.

Apart from this, many difficulties are encountered in this topic. Inferring authors from a novel or story is more difficult, as article samples are usually shorter texts. In addition, the articles may not be written in proper Turkish depending on the dialect or accent of the person. Such situations make it difficult to identify the author from the article. While there are many researchers who have developed using transfer learning methods in Turkish, these researchers have not made a comparison with machine learning algorithms and it has not been evaluated whether transfer learning is a necessary method for this topic.

### 1.3 Contribution

By bringing an innovation in this study, we collected our own data and prepared our own dataset, and then we were able to compare performance using machine learning and transfer learning methods. Rather than using only one method, the strengths and weaknesses of two different methods were compared. While there are many researchers who have developed using transfer learning methods in Turkish, these researchers have not made a comparison with machine learning algorithms and it has not been evaluated whether transfer learning is a necessary method for author identification.

In this study 7 machine learning algorithms and 2 transfer learning method has been used. Due to the low number of transfer learning algorithms and the difficulty of implementing their own tokenizer structures, only 2 were used. Apart from this, due to the limited number of algorithms that support Turkish language, BERT and GPT were preferred, algorithms with two different basic structures were preferred, and comparisons were made in these algorithms. Since machine learning algorithms are relatively easy to implement, many algorithm attempt have been made.

Depending on the number of authors, 3 different data sets were prepared and studies were carried out on these datasets. A dataset consisting of authors with the most articles was created and presented to developers on github. Anyone who wants can access this dataset and do their own work.

It is free access to this dataset will also lead to future Turkish studies. It is a dataset that can be used not only on this topic, but also in text mining applications such as sentiment analysis or finding the topic of the article etc. This is one of the most important contributions we have made.

Author identification is a multi-class classification problem that deals with labeling an anonymous text with one of the potential authors. We tested some transfer learning methods including Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT) and also some classical machine learning methods including Support Vector Machines (SVM), Gaussian Naive Bayes (GaussianNB), Multi-Layer Perceptron (MLP), Logistic Regression (LR), Stochastic Gradient Descent (SGD) and ensemble learning methods including Extremely Randomized Trees (ExtraTrees), and eXtreme Gradient Boosting (XGBoost) by python implementations using scikit-learn library. Additionally, we used the JPype python module to provide full access to Zemberek Java Library and the matplotlib and seaborn libraries for data visualization.

### 1.4 Organization of the thesis

This thesis contains 5 chapters organized as follows.

In chapter 2, We carried out a literature review. Since we use two different types of algorithms, we both researched the methods using machine learning on this topic, and also researched the transfer learning algorithms made on this topic. In line with the information we gained in these studies, we received assistance in the preprocessing we can do on our dataset or in the determination of the methods to be used.

In chapter 3, how the dataset was collected and the data in the dataset were defined. Brief descriptions of the algorithms were given and information was given about where these algorithms are used. Information was given about the configuration settings we use in these algorithms, and brief explanations were made about how to find the best parameters.

In chapter 4, the accuracy rate obtained in the algorithms were listed, our failures and achievements were explained, and our comparisons between the algorithms were mentioned. Opinions were given about the effect of our dataset on accuracy score. The effect of few or more authors on accuracy score was compared.

In chapter 5, we summarized our results and gave feedback about our future work.

# CHAPTER TWO
# LITERATURE REVIEW

In this section, we prepared a literature review from the last years about our topic. We will look not only at the articles on transfer learning topics, but also at the machine learning studies we will look at. While conducting our research, we conducted a literature review of articles written in recent years, since there are many articles published in recent years. We will look at the algorithms used, the techniques used, and all of the preprocessing steps used. We will conduct a detailed literature review.

## 2.1 Related work

The task of identifying authors has been studied in different languages for different purposes since 2000. An important part of the literature consists of studies on English language (Sari & Stevenson & Vlachos, 2018; Barlas & Stamatatos, 2020; Ramezani, 2021; Fourkioti, Symeonidis & Arampatzis, 2019). There are also many studies done in many different languages including Japanese (Okuno, Asai & Yamana, 2014), Mongolian (Damiran & Altangerel, 2014), Persian (Ramezani, Navid & Mohsen, 2013), Albanian (Paci et al., 2011), Indian (Pandian et al., 2016; Digamberrao & Prasad, 2018), Brazilian (Oliveira et al., 2013), Russian (Ramanov et al., 2020; Fedotova et al., 2022), German (Sage et al., 2020), and Arabic (Otoom et al., 2014). When the existing studies were examined, it was seen that different types of data sets were used for author identification tasks. Some studies have been carried out on newspaper articles, while others were carried out on poems, novels, email content (Vel et al., 2001), song lyrics (Kırmacı & Oğul, 2014), source codes (Bandara & Wijayarathna, 2013), or tweets, blog posts, and forums (Alonso-Fernandez et al., 2021). In some cases, different types of data sources were combined or compared (Atar, Esen & Arabaci, 2018).

Early studies in author identification focused on different stylometric techniques. These techniques are based on identification of style markers including lexical and character features or syntactic and semantic features that quantify writing style (Diri

& Amasyalı, 2003). The style markers can be exemplified as sentence length, function word and character n-gram frequencies, the number of verbs and punctuation marks in the sentences, vocabulary richness measures etc. With the development and widespread use of machine learning models over the last decade, machine learning-based author identification has become a promising solution for author identification. Mohsen et al. applied a deep learning method with name Stacked Denoising Auto-Encoder for extracting document features and then used the SVM classifier (Mohsen, El-Makky & Ghanem, 2016). They used a subset of RCV1 dataset, which contains 100 documents from each of the top 50 authors and reached classification accuracy up to 95.12% under different settings. In authorship attribution experiments were carried out using a Feedforward Neural Network model (FNN) and LR and 95.93% of accuracy was achieved on one of the four widely used datasets (Sari, Stevenson & Vlachos, 2018). In another recent study, pre-trained language models were applied in the field of author identification. They demonstrated that BERT and ELMo pre-trained models achieve the best results (as 92.86%) on a cross-domain dataset (Barlas & Stamatatos, 2020). Ramezani employed seven well-known classifiers by using the TF-IDF scheme on two English and Persian datasets and obtained 0.902 and 0.931 accuracy, respectively (Ramezani, 2021). Fourkioti et al. combined the three language models based on characters, words, and POS trigrams and achieved the best generalization accuracy of 96% on movie reviews (Fourkioti, Symeonidis & Arampatzis, 2019).

A few research on author identification have been carried out in Turkish language. Some studies are based on NLP techniques, while others are based on machine learning techniques. One of the first studies in this field was Diri and Amasyalı extracted 22 style markers for the 18 different authors to determine the author of an anonymous text (Diri & Amasyalı, 2003). They obtained a success rate of %84 on average. Örücü and Dalkılıç proposed two methods for determining the corresponding author of an anonymous text. Author-specific N-gram Method and Support Vector Machine (SVM) were applied to newspaper columns of 16 authors. The first method reached a success ratio of 87% with 1-grams while SVM had a success ratio of 77% with 2-grams (Örücü & Dalkılıç, 2010).

Atar et al. collected the columns from the electronic archives of two different newspapers and created a dataset containing 100 training and 20 test articles for each of the 237 authors. They trained the Word2vFisher and Doc2Vec models using a large corpus in Turkish and used the SVM classifier to classify the columns. They stated that the Skip-Gram approach is more successful when compared to the CBOW approach (Atar, Esen & Arabaci, 2018). Kuyumcu et al. used the same dataset in and applied the TF-IDF weighting method for the vector space that was a combination of word 1-3 n-grams and character 2-6 n-grams. They used Ridge Regression as a classifier and achieved an accuracy of 89.6% (Kuyumcu, Buluz & Kömeçoğlu, 2019).

Karaman et al. used a total of 1295 news articles of 10 different authors to predict unknown authors of an articles by using TF-IDF technique and Random Forest, Decision Tree, Naive Bayes and SVM algorithms. They achieved success rates 80%, 69%, 94% and 97% of F-measure, respectively (Karaman, Dalkılıç & Eker, 2020).

Mohsen et al. used deep learning method for n-gram analysis. They applied a Stack Denoising Auto Encoder for extract features. Their algorithm is SVM for classification. Their best accuracy rate is 95.12% (Mohsen, El-Makky & Ghanem, 2016).

One of the first articles I've seen that produces a solution on this issue using siamese network. Saedi & Dras produces a solution on this issue using siamese network. One of the first articles I've seen that. Siamese network usually using for graphical issues. If you have image or some graphics, you can easily implement your data. But they tried with natural language processing applications. They got 72% accuracy rate for english pan-15 dataset (Saedi & Dras, 2021).

Fabien et al. received very good performance on their dataset. They used Enron Email, Blog Authorship and IMDB datasets. Their accuracy rate on Blog authorship is 65.4%. Their base model uses the TF-IDF. When they started working on other datasets, they actually got very high scores there. For Enron dataset, their accuracy 99.95% and for IMDB dataset, their accuracy rate 99.6%. As the number of writers

increased, their performance decreased, but they still achieved very good scores. Their input token length 512, which is used the highest number of tokens BERT could provide (Fabien, Villatoro-Tello, Motlicek & Parida, 2020).

Polignano et al. works with Pan-19 dataset. Their purpose is, will a better result be obtained by using the transfer learning method from the machine learning algorithm that won the pan-19 competition? They want to test it, that's why they used BERT DNN. They configured 512 tokens and dropout value is 0.3. They stated that the BERT algorithm achieves better scores by a small margin. However, due to the working principles of transfer learning algorithms, they stated that the load on the system increased 37 times, since the processing times were very long (Polignano, Gemmis & Semeraro, 2020).

Barlas et al. preferred a dataset written in English. Multiple transfer learning algorithms are used, these are BERT, ELMo, GPT-2, ULMFiT. They preferred to run 5-10-20 epochs. Apart from the author recognition, they also separated the texts by topic. Their scores are average 83.07% for BERT, 84.52% for ELMo, 70.37% for GPT-2 and ULMFiT. In their study, they achieved lower performance in separating texts according to the topic than in identifying the author. Therefore, they obtained the best scores in the author identification process (Barlas & Stamatatos, 2020).

Fávero said that by making differences in the layers in the base BERT model, they achieved an average of 13% performance increase on their dataset. This is the first research article I've seen making changes to the base model. After the fine tuning process, the training phase of the base model took a few hours on a smaller dataset (Fávero & Casanova, 2021).

Devlin et al. works with BERT and GPT algorithm. He said BERT is simple to implement and works great. He tried fine tuning approach BERT large and BERT base, but no difference was observed between the two approaches. They achieved similar success rates with ELMo and GPT algorithms (Devlin, Chang, Lee & Toutanova, 2019).

It can be noticed from the above examples that the author identification studies in Turkish language are open to development. In this paper, we introduced an author identification method using some classical machine learning and transfer learning methods. We also investigated the classification performance of the selected techniques on three different sized author groups and two distinct n-gram profiles.

# CHAPTER THREE
# MATERIALS AND METHODS

## 3.1 Dataset

In this study, the dataset was gathered by us from plenty of Turkish News Websites. The collected articles include independent topics written by the authors and it includes articles written from 2005 to the present. Dataset contains 86,852 articles, 49 authors and an average of 1772 texts per author. In order to experience the performance of the algorithms, we created 3 datasets and according to the author who has the most articles while creating these datasets. The maximum number of articles written by an author is 3495.

The information about the top 20 authors is given in Table 3.1. One of our authors has a total of 2391 articles and the average number of words she/he used in these articles is 849.54. Additionally, we also see the minimum average word count per article is 335.11. While the maximum total word count for an author is 2,031,274 the average word count per article is approximately 445. No preprocessing or filtering was applied to the original corpus. Some of the texts have large spaces, URL links, special characters and full capital words.

Table 3.1 Information about the top 20 authors

| Description | Author-10 | Author-15 | Author-20 |
|---|---|---|---|
| Count | 30372 | 43451 | 55108 |
| Mean | 456.53 | 416.92 | 445.85 |
| Std | 158.54 | 150.04 | 192.01 |

In this table count represents number of non-empty values, mean is the average value, std is the standard deviation. As can be seen in this table, we have a sufficiently large dataset.

Let’s take a look at the technical data in Table 3.2 about our dataset.

Table 3.2 Describe words on 10 authors dataset

| Description | Author-10 | Author-15 | Author-20 |
|---|---|---|---|
| Min words | 4 | 4 | 3 |
| 25% words | 329 | 323 | 334 |
| 50% words | 377 | 372 | 396 |
| 75% words | 481 | 471 | 495 |
| Max words | 3111 | 3111 | 3651 |

In this table count represents number of non-empty values, mean is the average value, std is the standard deviation, min is the minimum value, 25% is the 25% percentile, 50% is the 50% percentile, 75% is the 75% percentile, max is the maximum value. Percentile meaning is how many values are fewer than the specified percentile.

We can visualize our words using WordCloud in Figure 3.1, it means that as the size of the words increases, the frequency of use also increases. Let's also take a look at the words in the dataset visualized as a cloud or different shapes. Instead of using numeric values, it may be more useful to use such visual elements.


Figure 3.1 Wordcloud created with author-20 dataset

Let’s give some information about how the data is collected. In our thesis, only the C# programming language was used in the data collection phase, while the python language was used for all other operations.

The digital text articles of the national newspapers that provide online broadcasting services were downloaded by using the developed software using C# programming language in the Visual Studio.NET development environment in Figure 3.2

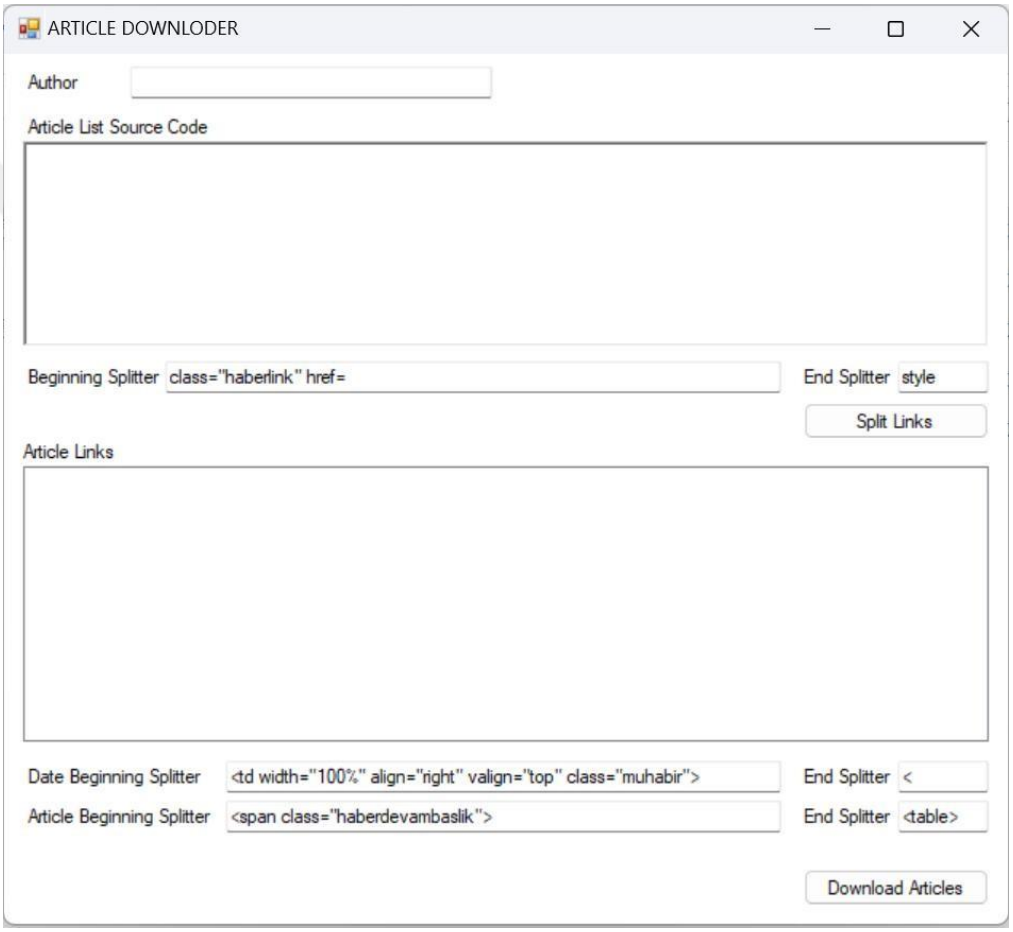


Figure 3.2 Graphical User Interface developed to download newspaper articles

The web links used to access newspaper articles follow a certain pattern. Using this pattern, the links containing the articles of the selected authors were automatically collected. Then, the source code of the web pages in these links was gathered using the HttpWebRequest and HttpWebResponse classes. The resulting text content also includes Javascript and HTML codes. The articles were cleaned from these source

codes and stored on the local disk as pure text with the unique names as "authorname_data.txt".

If we take a look at the unigram and bigram analyzes used by our authors in groups of five, we can have information about the topics of the articles written by the authors. Let's view the unigram and bigram analysis of our authors from one to five in Table 3.3.

Table 3.3 Unigram and bigram analysis of authors from 1 to 5

| **Author No** | **Unigrams** | **Bigrams** |
|---|---|---|
| Authors 1 | gürüz, dershane, kontenjan, yükseköğretim, öss, fen, rektör, ösym, kolej, mezun, öso, veli, okul, lise, öğretmen, özet, sınav, | ol üniversite, hemen herkes, eğitim sistem, lise mezun, sınav gir, bin öğrenci, yabancı dil, vakıf üniversite, pek çoğu, mezun ol, yok başkan, eğitim bakan, özel okul, eğitim bakanlık, |
| Authors 2 | düşük, emtia, toni, fiyatla, borsa, endeks, gerile, artırım, enflasyon, negatif, pozitif, çeyrek, likidite, cari, küresel, yüzde, baz, | ilk çeyrek, hisse senet, faiz oran, para politika, faiz artır, faiz düş, değer kaybet, faiz art, çift hane, değer kayıp, art yüz, kur art, para genişle, yüzde art, faiz artırım, küresel kriz, küresel |
| Authors 3 | çankaya, refah, chp, derviş, dek, derya, lider, fazilet, politbüro, 2000, meclis, gözle, kurultay | demokrat, ön sür, parti meclis, düş kırık, buse yönetim, yol aç, abd yönetim, merkez sağ, yüce divan, seçim git, körfez savaş |
| Authors 4 | katalog, fuar, senfoni, eleştirmen, bellek, opera, yayınevi, roman, müze, orkestra, hizlanin, eser, | bu kitap, kültür merkez, bir şehir, oku bir, oku gerek, nazım hikmet, yaşar kemal, ilgi çek, kitap bir, turizm bakanlık, bir yazar, kültür sanat, bir |
| Authors 5 | serinkanlı, mesela, egoizm, belagat, acaba, samimi, duygu, ben, yani, pespaye, gazete, şu, mülakat, iftira, | bir ifade, köşe yazar, bir gazete, iş neden, birinci sayfa, yayın yönetmen, bir cevap, yazı iş, kendi sor, bir kere, bir insan, genel yayın, ilginç bir, şu |

If we look at our first 5 authors, we can understand that there are articles written on various topics such as education, art, politics, and economy. This already shows the diversity in our dataset. Continue with authors 6 to 10 in Table 3.4.

Table 3.4 Authors from 6 to 10 unigram and bigram analysis

| Author No | Unigrams | Bigrams |
|---|---|---|
| Authors 6 | mevzu, serenay, harika, akal, pekkan, hande, eğlen, cihangir, tıklım, ınstagram, demet, kadın, albüm, alaçatı, sene, magazin | güzel bir, doğum gün, bin il, kadın ol, sabah kadar, ve tabii, bir erkek, bir kız, bir mekan, sezen aksu, yeni nesil, hülya avşar, dans et, bir kadın, şarkı söyle, sahne çık, demet akal, sosyal medya |
| Authors 7 | Kulüp, yolla, fenerbahçe, fatih, otomobil, aysal, rezalet, teke, corona, adam, falan, şahane, galatasaray, habertürk, zannet | hemen hemen, sosyal medya, teknik direktör, emin ol, bir biçim, haber yap, anla kadar, ben sor, ol anla, hele hele, ben göre, aziz yıldırım, büyük bölüm, büyük ihtimal, tam aksine, |
| Authors 8 | cumhurbakan, keza, öneri, mmkn, tutum, erdoğan, genelkurmay, başbakan, zer, lider, kktc, koşul, | devlet bakan, kıbrıs konu, bu neden, yanıt ver, silah kuvvet, ilgili ol, esas al, şöyle özetle, başbakan yardımcı, milli güvenlik, güvenlik kurul, |
| Authors 9 | sezi, izlenim, ihtilal, yöre, simgesel, gene, harikulade, varoğlu, sağduyu, hadise, yazgı, sözgelişi, satır | silah kuvvet, hükümet kur, türkiye abe, sınır öte, yol harita, fatih terim, yan sıra, güvenlik güç, özen göster, bu karşın, bu hükümet, siyasi parti |
| Authors 10 | ucuz, imkan, lira, dolar, katrilyon, gelir, teyze, öde, mevduat, banka, dolay, sat, faiz, ytl, bono, fiyat, anlatım, üretim, döviz | bin ton, dayalı ol, milyar ytl, fiyat art, mal hizmet, yüzde oran, üretim art, faiz öde, sat al, katrilyon lira, yatırım üretim, türk lira, dolar fiyat, yüz oran, ayşe hanım, döviz açık, hanım teyze, |

When we examine our authors above, we encounter sports terms that we have observed for the first time. We can observe that we have authors for almost all topics.

While this increases the word variety, it will cause the algorithm to receive a low accuracy rate if it does not work properly. Continue with our authors the number 11 and 15 in Table 3.5.

Table 3.5 Authors from 11 to 15 unigram and bigram analysis

| **Author No** | **Unigrams** | **Bigrams** |
|---|---|---|
| Authors 11 | ihtimal, çıta, kürt, birlik, kıbrıs, gitgide, deyiş, klişe, siyaset, inandırıcı, reform, ray, barış, yalnız, siyasal, istikrar, demokrasi, cemal | ama aynı, öte beri, bunca yıl, bu açı, çare yok, soru işaret, şöyle de, başka çare, kürt sorun, türkiye avrupa, kıbrıs çözüm, ağır bas, ab yol, koalisyon hükümet, avrupa birlik, hukuk devlet |
| Authors 12 | hani, dalkavuk, adam, fıkra, rahmetli, eşek, sen, kayra, herif, mecek, deyim, muhterem, laf, memleket, paşa, yağdanlık | bu ki, bu de, telefon et, değil mi, sayın bakan, geçen gün, laf et, kurtul savaş, ne de, mi ali, el git, avrupa ol, de de, şöyle de, de mi, ismet paşa, böyle de, bil mi, bir laf, ki mi |
| Authors 13 | fidan, aforizma, ayhan, anakent, arif, efendi, sütun, arıcıoğlu, birşey, fıkra, bilvesile, vs, hortum | yanıt ver, uğur mum, trilyon lira, milyar lira, chp milletvekili, mesut yılmaz, istanbul belediye, genel müdür, tayyip erdoğan, bir dost, mesut yıl, |
| Authors 14 | şirket, ciro, böylelikle, turkcell, gerekse, trap, arçelik, meral, zira, müşteri, okur, peşpeşe, tüket, siemens | grup başkan, al bilgi, benzeri bir, son dönem, kafa yor, nükleer santral, halk ilişki, geri öde, bin lira, bilgi göre, lira öde, yönetim kurul, kurul başkan, genel |
| Authors 15 | başkanvekili, ilişkin, akşener, anımsa, süreç, parti, suriye, kobani, chp, sohbet, zemin, ittifak, idlib, davutoğlu, algı, uzlaşı, hdp, | soru yönel, alt çiz, tbmm başkan, bu aşama, ara bulun, dil getir, cumhurbaşkanı erdoğan, parti yönetim, son dönem, grup başkanvekili, bu dolayı, vurgu yap, yakın geçmiş, parti, ak parti |

We also come across topics such as daily news, politics and economy in these authors. Although it is observed that similar terms are used in topic of politics, different terms stand out in articles written on other topics. Continue with the our authors the number 16 and 20 in Table 3.6.

Table 3.6 Authors from 16 to 20 unigram and bigram analysis

| **Author No** | **Unigrams** | **Bigrams** |
|---|---|---|
| Authors 16 | temas, kıbrıs, klerides, camia, çeşit, ab, başlıca, terörizm, giriş, analist, cohen, ilinti, abi, müzakere, argüman, sürtüş, diplomat, deyiş, diplomasi | bölge ülke, son zaman, ön sür, bu bakım, dışişi bakan, rus taraf, türkiye ab, bir tavır, ilk bak, bir rol, bu bağlam, dış politika, türk yunan, buse yönetim, müzakere süreç, türk taraf, diğer bir, bir deyiş, uluslararası camia, türk diplomasi |
| Authors 17 | apple, amerika, global, dijital, york, ypg, vasat, trend, konsantre, amerikan, hayli, tamamen, seküler, gündelik | üzeri düşün, büyük ihtimal, insan ol, kadar fazla, bu yüz, son derece, ol düşün, bir tane, var ol, 21 yüzyıl, bir insan, ne yok, imkan ol, ne york, beyaz saray, hayat tarz, ilk önce, ulusal |
| Authors 18 | modern, evvela, bağnaz, merkeziyet, radikalizm, milliyetcomtr, sosyolog, metot | siyasi bir, bir siyasi, ki hem, koç de, bir hukuk, sebep ol, bir sosyal, bir toplum, ismet paşa, ana dbp, sağ sol, ekonomik sosyal, son derece |
| Authors 19 | rant, firma, eminönü, trakya, belde, kadıköy, tekirdağ, ihale, topbaş, akli | daire başkan, bir okur, yönetim kurul, başlık yazı, ne yazık, aday ol, genel başkan, genel Merkez, büyükşehir belediye, belediye başkanı |
| Authors 20 | çuvalla, deney, onbir, bryson, aniki, tv, nutuk, elli, çalkala, çarpı, birbuçuk, bağlantı, yirmi | dünya yer, bu karşılık, on yıl, insan hak, tek bir, ki hem, ab ülke, şu an, bir milyon, zaman zaman, avrupa konsey, geri çevir, örnek var, müthiş bir |

Thus, we have completed the unigram and bigram analysis of all authors in our dataset. Looking most correlated unigrams and bigrams give us to information about which word patterns the authors use more often. Although the names of the authors are given secret for this thesis in Figure 3.3.

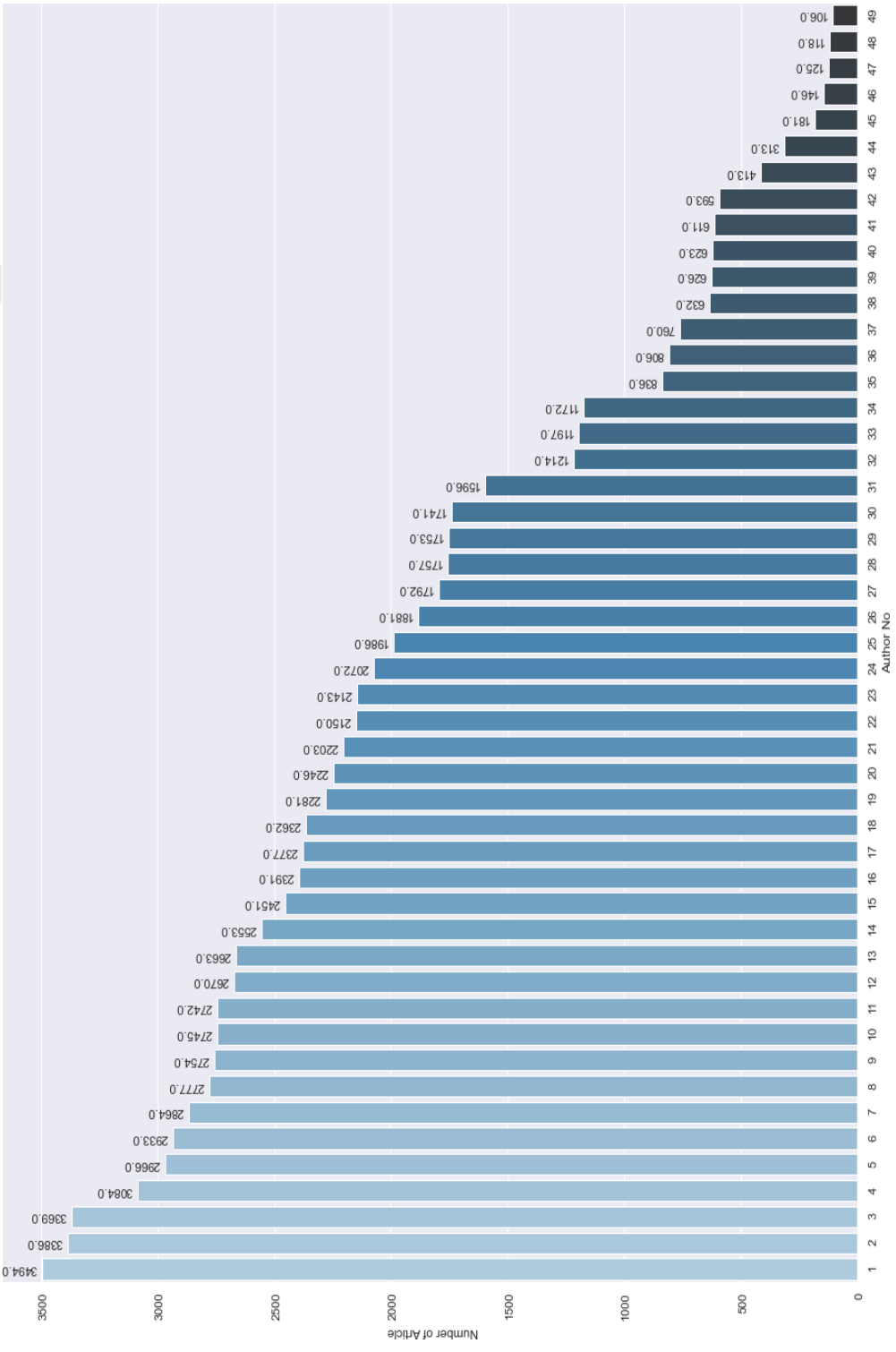


Figure 3.3 Number of articles for each author

Brief information given about words in Table 3.7 and Table 3.8.

Table 3.7 Total article and total word count

| Dataset | Total Article | Total Word Count |
|---|---|---|
| Dataset-10 | 30372 | 12770000 |
| Dataset-15 | 43451 | 17861840 |
| Dataset-20 | 55108 | 24218293 |

Table 3.8 Article, total word and average word count per article

| Author | Article Count | Total Word Count | Average Word Count Per Article |
|---|---|---|---|
| Author 1 | 3494 | 1349874 | 386.34 |
| Author 2 | 3386 | 1161662 | 343.07 |
| Author 3 | 3369 | 1093380 | 324.54 |
| Author 4 | 3084 | 1562320 | 506.58 |
| Author 5 | 2966 | 1101166 | 371.26 |
| Author 6 | 2933 | 1274944 | 434.68 |
| Author 7 | 2864 | 1733088 | 605.12 |
| Author 8 | 2777 | 1273953 | 458.75 |
| Author 9 | 2754 | 951911 | 345.64 |
| Author 10 | 2745 | 1267702 | 461.82 |
| Author 11 | 2742 | 953534 | 347.75 |
| Author 12 | 2670 | 1017759 | 381.18 |
| Author 13 | 2663 | 960657 | 360.74 |
| Author 14 | 2553 | 1338523 | 524.29 |
| Author 15 | 2451 | 821367 | 335.11 |
| Author 16 | 2391 | 2031274 | 849.54 |
| Author 17 | 2377 | 1086461 | 457.07 |
| Author 18 | 2362 | 989805 | 419.05 |
| Author 19 | 2281 | 974750 | 427.33 |
| Author 20 | 2246 | 1274163 | 567.30 |

We can also look at the number of words that we encounter the most in our entire dataset in Table 3.9. No preprocessing steps were applied before preparing this table. Therefore, most of the words here are the words called stop words in Turkish. Generally, the most used words are removed from the datasets in order to increase the distinguishing features of the words and to reduce the processing load. We have done this. While some researchers do not remove the most used words from the dataset, they perform the removal of the words in that library through ready-made libraries in python. While this removes all stopwords for english language it doesn't work well for turkish.

Table 3.9 Most used words

| **Word** | **Times** |
|---|---|
| Bir | 645035 |
| Ol | 584202 |
| O | 408151 |
| Bu | 289710 |
| Et | 236446 |
| Yap | 219041 |
| De | 169344 |
| Türkiye | 145186 |
| Ver | 141802 |
| Var | 131841 |
| Yıl | 126636 |
| Al | 125508 |
| Gel | 116169 |
| Ben | 103663 |
| Değil | 95947 |
| Ara | 89742 |
| Kadar | 88820 |
| Çık | 87447 |
| Sonra | 87133 |
| Kendi | 86095 |

### 3.2 Preprocessing

The original data can not be sent to a machine learning model or transfer learning model. Because real world data which is collected from websites often noisy, incomplete or misspelled. If we give the text directly to the algorithms, it will be very difficult to get the result we want and we will have to achieve low performances. That's why preprocessing must be always most significant part for us. The same preprocessing steps were made for both machine learning and transfer learning algorithms and as a result a common dataset called author-10, author-15, author-20 was created for algorithms. If we briefly touch on what kind of operations we do during the preprocessing stage, these are, removing URL links, normalizing text to lowercase, filtering stopwords, filtering special characters, removing digits, white space formatting. Let's briefly talk about what processes are done in these stages.

Removing URL links, since our dataset contains content downloaded from the internet, links to the site from whichever site was downloaded were added at the end of flies during the data acquisition phase. Since these links are not meaningful data for our algorithms, we cleaned the links from the text.

Normalizing text to lowercase, This process, which is frequently used in applications related to data mining in general, has been converting into lower case by the algorithm in order to prevent the words from being misunderstood even though the spelling of the words originating from the use of upper or lower case letters is the same and also helps doing parsing.

Filtering stopwords, It is the process of removing word groups such as conjunctions and prepositions that are not meaningful on their own from the texts. The aim is to minimize the effects of these words on the performance by entering the algorithms and provide more accurate estimation.

Filtering special characters, the process of removing special characters such as punctuation marks and emojis in our dataset has been performed.

Removing digits, numeric values and datetime removed from our dataset. Although this process is done for every language, some developers find date and numeric data important and convert them from numeric values to text expressions. This is also one of the methods that can be followed.

White space formatting, In cases where extra spaces are left between words, horizontal tab, newline, carriage return are removed. So that all the words were sorted for each text of the authors. The source code of these steps we have performed is given in Figure 3.4.

```
def special_characters (text):
    punctation = string.punctuation
    return text.translate(str.maketrans("","",punctation))


def stop_words_clean (text):
    words = set(stopwords.words("turkish"))
    return (' '.join([i for i in text if i not in words and not i.isnumeric()]))

def remove_html(text):
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)

def remove_URL(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',text)

df['special_characters']=df['content'].apply(lambda x : remove_URL(x))
df['special_characters']=df['special_characters'].apply(lambda x : remove_html(x))
df["special_characters"] = df["special_characters"].apply(lambda x: special_characters(x))
df["special_characters"] = df["special_characters"].str.split()
df["stop_word"] = df["special_characters"].apply(lambda x : stop_words_clean(x))
df= df.applymap(lambda s:s.lower() if type(s) == str else s)
```

Figure 3.4 The source code of some preprocessing steps

After these stages were completed, we proceeded to the process of finding the root of the word, which we usually encounter in natural language processing applications. There are two different options that we can use here Zemberek and a library called snowballstemmer in Python. Snowballstemmer has a import TurkishStemmer. Actually when we look at the studies and documentation on the github page, we observed that although it seems to produce good results in the examples given, many words have difficulty in finding their roots an deven cannot divide them meaningfully.

Zemberek is one the best java tools for Turkish Natural Language processing. With languages such as Python, Ruby, Javascript and more, you can easily do natural language processing for Turkish.

A java virtual machine must be created in Python to use Zemberek. Afterwards, the zemberek-full.jar file of zemberek should be given as path. Then java virtual machine should be started via jpype library. In Figure 3.5 you can check the source code of Zemberek implementation.

```
def Lemmatization(sentence):

    analysis: java.util.ArrayList = (morphology.analyzeAndDisambiguate(sentence).bestAnalysis())
    token = sentence.split()

    pos=[]
    for index,i in enumerate(analysis):
        if str(i.getLemmas()[0])=="UNK":

            pos.append(token[index])
        else:
            pos.append(str(i.getLemmas()[0]))
    return pos


TurkishSpellChecker: JClass = JClass('zemberek.normalization.TurkishSpellChecker')
spell_checker: TurkishSpellChecker = TurkishSpellChecker(morphology)
def spellChecker(tokens):

    for index,token in enumerate(tokens):
        if not spell_checker.check(JString(token)):
            if spell_checker.suggestForWord(JString(token)):
                tokens[index] = spell_checker.suggestForWord(JString(token))[0]

    corrected = [str(token) for token in tokens]

    return " ".join(corrected)
X = []
x = table.stop_word
for i in x:
    text = re.sub('\W+', ' ', str(i))
    text = spellChecker(text.split())
    lemmaList = Lemmatization(text)
    text = " ".join(lemmaList)
    X.append(text)
```

Figure 3.5 The source code of zemberek implementation

We did spell check, lemmatization steps with Zemberek. It is useful for turkish natural language applications.

What else we can do with Zemberek? It has 9 modules. You can make turkish morphological analysis, word generation, turkish tokenization and sentence boundary detection. You can examples with module named zemberek-examples. You can create console applications. It provides a language model compression algorihm. You can make text classification based on java port, turkish named entity recognition, basic spell checker, word suggestion and noist text normalization.

We firstly used spell checker, If there is an error in the spelling of the word, it enters the if loop and returns the correct version of the word. Second we did lemmatization, it is based on morphological analysis of words and helps to find root of words. Since lemmatization can be done in Zemberek, we chose this direction. After all these stages, we have now prepared a proper dataset from our articles.

### 3.3 N-gram

N-gram models are often applied in domains such as statistical natural language processing, computational linguistics, linguistic modeling, statistics, and communication theory nowadays.

N-gram models are popular because they are scalable and considerably simpler than other methods and models that provide the same function. N-gram is used in a variety of natural language processing techniques. The word prediction tool (predictive text) and autocorrect, which we notably use when sending phone texts, benefit of n-grams. For example, our phone keyboards that suggest the word "me" after we type the word “call”. It uses statistical data and n-grams when making this suggestion.

The n in the n-gram expression represents the degree of repetition. How many groups we will classify the words in is hidden in the expression n. 1-gram is called unigram. 2-gram is called bigram. 3-gram is called trigram and if the number is greater than 3 it is simply called n-grams. Let’s elaborate n-gram by giving an example.

“Bu bir örnek cümledir”. Our unigrams for this sentence bu, bir, örnek, cümledir. Our bigrams for this sentence, bu bir, bir örnek, örnek cümledir. Our trigrams fort his

sentences, bu bir örnek, bir örnek cümledir. It is possible to use larger n-grams for sentences containing more words. However, as the number of n-grams increases, the word groups accumulated in the pool will also increase, so the effect on performance should be observed.

### 3.4 TF-IDF vectorizer

In this study, The TF-IDF vectorizer has been used to explore similarity between text documents. This is a very common algorithm for converting text to a meaningful representation of numbers, also known as vector representation to feed into machine learning algorithms. It is easy to implement and works fast. TF-IDF was used in the early 1970s to solve an information retrieval problem and then has been successfully used in document classification, topic modelling etc. TF represents the number of times each word occurs in text, article or any kind of datasets. For example, if the word "save" occurs 20 times in a text and the entire text has 1000 words, the TF value is 0,02 (20/1000). IDF shows the importance of the word "save" for a text. It is obtained by dividing the total number of texts by the number of texts containing the term. A score closer to zero indicates that the word is used more often.

TF-IDF Vectorizer in scikit-learn library takes ngram_range (1,1) as default parameter. The parameter (1,1) means only unigrams, (2,2) means only bigrams, (1,2) means using both unigrams and bigrams, are used to create TF-IDF vectors. The larger n value indicates a larger probability pool. As the number of n-grams increase, we will see the decrease in the accuracy score in the experimental studies section.

### 3.5 Algorithms

In this study we worked with best known machine learning algorithms and transfer learning methods to compare the power of transfer learning methods. In shortly, transfer learning is when an algorithm trained for a different problem, storing this information and later using it to solve another problem.

Using transfer learning methods allows researchers and developers to achieve good results even If their data is scarce. Therefore, there are different transfer learning algorithms and structures used in different fields.

BERT and GPT algorithms are the transfer learning algorithms we prefer, which are most suitable for our topic and dataset. Because the data we had Turkish articles collected from Turkish news websites. Therefore, the transfer learning method we will use had to have support for the Turkish language.

Apart from these, transfer learning methods with corpus belonging to different languages were used and very low accuracy performance were obtained due to the differences in the trained word dataset. Therefore, only these 2 methods were used in our study.

### *3.5.1 Machine learning algorithms*

Support Vector Machine is one of the supervised learning methods generally used in classification problems. Basically, it tries to separate two classes with a line or plane. It also makes this separation according to the elements at the boundary.

Naive Bayes is a probabilistic machine learning model used for classification problems. It can do good work with small data. Naive Bayes assumes that each class follows a Gaussian distribution and NB is a generative model.

Multi Layer Perceptron has emerged as a result of the studies done to solve the XOR Problem. It works effectively especially in classification problems.

Linear Regression is a popular, uncomplicated and a supervised learning algorithm. It is the simplest form of regression that is also used to examine the mathematical relationship between variables.

Stochastic Gradient Descent is a linear classifier such as SVM or Linear Regression and has been successfully applied large data sized datasets. It is easy to implement and has many possibilities in code tuning.

Extra Trees are consisting from bunch of decision trees It is also easy to use and set various hyperparameters. It gives better performance than the Random Forest (RF) algorithm.

XgBoost is a algorithm that has been used frequently in competitions and researchers because it has achieved high success rates recently and we can say that it is very successful for our dataset.

### *3.5.2 Transfer learning methods*

Bidirectional Encoder Representations from Transformers known as Google BERT is an algorithm that is actively used for searches on its own website and has made our searches on google more meaningful and accurate recently. At the same time BERT is also used predict the next word in searches on Google. While trying to find the missing word, BERT examines the words that come after a word frequently and at the same time examines them as sentence integrity. Since it does these two reviews, it is one of the algorithms that really gives the best results in making the most accurate next word prediction.

BERT is trained with words in Wikipedia (that's 2,500 million words) and Book Corpus (800 million words). The words on these websites have created a huge pool of words for BERT.

BERT has support over 70 languages worldwide. After March 2020, smaller BERT models were released under the name of DistilBERT. DistilBERT is lighter version of BERT model and runs 60% faster and offers 5% performance loss. If you are working with huge dataset, you may ignore this performance loss.

For Researchers and developers BERT can use in sentiment analysis, text summarization, text generation and more. It is free to use and easy to implement your code. BERT or another transfer learning methods are not working with 10 or more epochs. It is usually enough 4 or 5 epochs. Because after the fourth or fifth epoch, there is no increase in performance, but since the performance remains at the same level and the training cost of an epoch is already very high, increasing the number of epochs will be unnecessary. You can check from Figure 3.6 how BERT layers work.

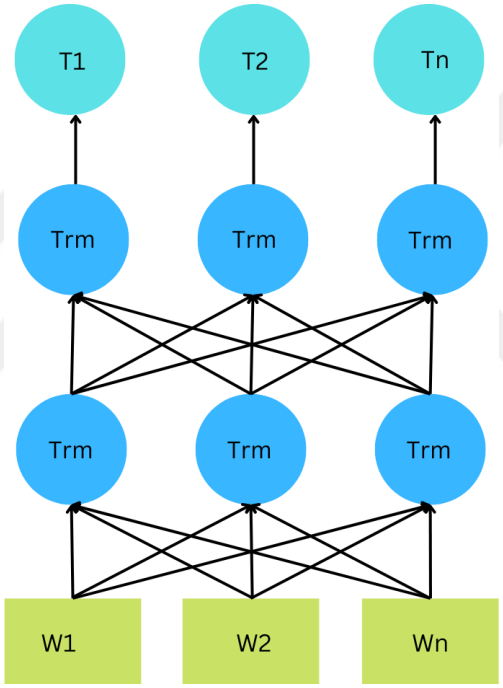


Figure 3.6 Bert model representation

The arrows indicate the information flow from one layer to the next layer. Final layer represents each input word. As we can see BERT is bi-directional. Bi-directional means BERT learns information both left and right side.

Intuitively, a deep bidirectional model appears to be more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model.

Generative Pre-Trained Transformer, It is a deep learning based natural language processing algorithm developed by OpenAi. This company based in San Francisco and works with artificial intelligence. Recently, many applications based on this algorithm have been made and these programs are generally chat applications.

OpenAi company uses ChatGPT, which was developed as GPT sub-base, benefits from the answers of the developers who develop with ChatGPT API. The answers given by this chat bot are collected by OpenAi and used in later development stages. In Figure 3.7 you can check the GPT model representation. It has an uncomplicated structure that is slightly different from the BERT model.

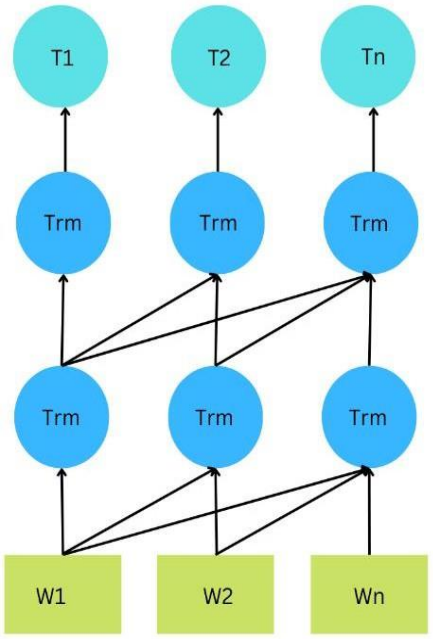


Figure 3.7 Model representation of gpt

Previously we saw the BERT was bi-directional. But GPT is unidirectional. What is unidirectional mean? Information flow of GPT is only from left to right. That is also provides to us consume less time for training. Because comparing with BERT, it has less information flow.

GPT's dataset has 410 billion filtered data. These data collected from internet, books and Wikipedia. GPT usage areas are natural language processing, production of pictures from words, chatbots that respond in a meaningful way.

GPT was created by blending the Long Short Term Memory and Transformer based architecture. The GPT model can be fine tuned. It is also can learn complex problems and patterns. That's why GPT and BERT models are selected for Transfer Learning for our dataset.

### 3.6 Proposed Models

We tested several combinations of hyperparameters' values by using Exhaustive Grid Search technique supplied by scikit-learn library and selected combinations with maximal classification accuracy. Final classification experiments were performed using selected hyperparameters for each classifier as given in Table 3.10. We used XgBoost and GaussianNB algorithms with default hyperparameters. Two different settings of the SVC classifier are employed to see performance comparison of different kernel functions.

Table 3.10 Hyperparameters of machine learning algorithms

| **Classifier** | **HyperParameters** |
|---|---|
| XgBoost | Default |
| SGD | Alpha=1e-05, max_iter=50, |
| LR | Max_iter=1000, solver=lbfgs |
| SVC-1 | Kernel=linear, gamma=scale, c=0.025 |
| SVC-2 | Kernel=rbf,gamma=2,c=1 |
| MLP | Alpha=1,max_iter=1000 |
| Gaussian NB | Default |
| Extra Trees | n_estimators=100,max_depth=1000, min_samples_split=2 |

The hyperparameters we have mentioned in the table above represent the values that work best for us. We can also check the best hyperparameters for transfer learning methods in Table 3.11.

Table 3.11 Hyperparameters of transfer learning methods

| Algorithm | Hyperparameters |
|---|---|
| **BERT** | Optimizer= AdamW, lr =5e-5, eps=1e-8, batch_size=32 |
| **GPT** | Hidden_size=768, max_seq_len=128, lr=1e-5 |

Whether it is a machine learning algorithm or a transfer learning algorithm, each algorithm has its own structure. Therefore, they have different hyperparameters of their own. These hyperparameters are already assigned a value by default, but these default values may not always work best for our dataset. That's why a method called GridSearchCV needs to be made. We used GridSearchCV is used for adjust the hyperparameters and after these step best one is selected. What is GridSearchCV?

GridSearchCV is a scikit-learn library that helps to optimize hyperparameters for algorithms. Although it is not preferred by some researchers because it takes a long time to find the best hyperparameter, these steps must be done in order to obtain the highest performance. As we saw in the Figure 3.8, code example of Xgboost algorithm. These code helps us to find best hyperparameters. Assuming that the value we will give for max_depth is 3 under normal conditions, it will not calculate other depth values that will only draw conclusions from 3.

We also use a method called cross validation to prove that the data we get from gridsearch is more accurate. GridSearchCv, which searches for the best parameters for us by cross validating 5 times, costs us time and consumes resources. Let’s look at how the values are set for the Xgboost algorithm, one of the most successful machine learning algorithms? Then we can continue with the code samples.

In Figure 3.9 we will see the best parameters and in Figure 3.10 we will see the results. In Figure 3.11 we will see the confusion matrix of xgboost algorithm.

```
import xgboost as xgb
parameters = {
    'max_depth': [3, 5, 7, 9],
    'n_estimators': [5, 10, 20, 50, 100,1000],
    'learning_rate': [0.03, 0.05, 0.1]
}

model_xgb = xgb.XGBClassifier(
    random_state=SEED,
)

model_xgb = GridSearchCV(
    model_xgb,
    parameters,
    cv=5,
    scoring='accuracy',
)

model_xgb.fit(x_train, y_train)

print('-----')
print(f'Best parameters {model_xgb.best_params_}')
print(
    f'Mean cross-validated accuracy score of the best_estimator: ' +
    f'{model_xgb.best_score_:.3f}'
)
cross_valid_scores['xgboost'] = model_xgb.best_score_
print('-----')
```

Figure 3.8 Code example of xgboost algorithm

```
model_xgb.best_estimator_
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=16,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=42, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

Figure 3.9 Best estimator of xgboost algorithm

```
model_xgb.cv_results_
```

```
{'mean_fit_time': array([426.40376811, 498.18318963, 401.21531444, 456.96322408]),
 'std_fit_time': array([5.88869296, 1.17392377, 4.47576024, 9.59843665]),
 'mean_score_time': array([0.07274923, 0.08917246, 0.07583394, 0.08452168]),
 'std_score_time': array([0.00598955, 0.0076416 , 0.00445891, 0.00204236]),
 'param_learning_rate': masked_array(data=[0.1, 0.1, 0.3, 0.3],
              mask=[False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_max_depth': masked_array(data=[6, 7, 6, 7],
              mask=[False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_n_estimators': masked_array(data=[100, 100, 100, 100],
              mask=[False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 100},
  {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100},
  {'learning_rate': 0.3, 'max_depth': 6, 'n_estimators': 100},
  {'learning_rate': 0.3, 'max_depth': 7, 'n_estimators': 100}],
 'split0_test_score': array([0.93598964, 0.9335443 , 0.95397008, 0.95210012]),
 'split1_test_score': array([0.92951669, 0.93383199, 0.95353855, 0.95325086]),
 'split2_test_score': array([0.93512658, 0.93498274, 0.95814154, 0.9523878 ]),
 'split3_test_score': array([0.93570196, 0.93627733, 0.95526467, 0.95512083]),
 'split4_test_score': array([0.93138665, 0.9325374 , 0.95181243, 0.94850403]),
 'mean_test_score': array([0.9335443 , 0.93423475, 0.95454545, 0.95227273]),
 'std_test_score': array([0.00260829, 0.00128464, 0.00211014, 0.00215938]),
 'rank_test_score': array([4, 3, 1, 2])}
```

Figure 3.10 Results of xgboost algorithm

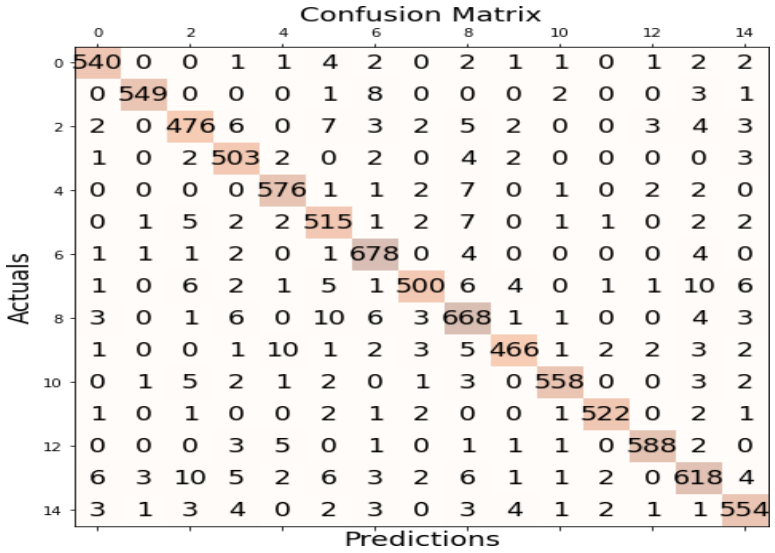


Figure 3.11 Confusion matrix of xgboost algorithm

# CHAPTER FOUR
# RESULTS AND DISCUSSION

In this study we worked with pandas, NLTK library and tensorflow, for drawing tables seaborn and matplotlib. All operations have been performed on laptop with Intel I7 11800H processor, RTX 3060 graphics card, 16 GB DDR4 Ram and Windows 10 operating system. However, since the RAM and graphics card memories required for the implementation of transfer learning methods such as BERT and the running of epochs were not sufficient, our operations were completed on Google Colab Pro with 80gb ram and Nvidia a100 graphics card.

Author identification performance of the classifiers are shown in Table 4.1. Accuracy was used as the evaluation metric to measure authorship identification performance. Let's talk about firstly with machine learning algorithms. When the performances of the classifiers are evaluated, it is seen that the most successful method for unigrams is SGD. This is followed by the XgBoost and LR methods with almost similar performance. SVC-2 (with rbf kernel) and Extra Trees classifier were also successful by showing over 90% performance. Considering the bigrams, LR and SGD performed better than other classifiers.

The accuracy score is going down with the increasing author count in experiments. Because it is getting harder to predict the right author in a larger pool of authors. Increase in the number of authors also increases the number of tags to be predicted and causes a performance decrease of approximately 4%.

We worked with 1-grams and 2-grams separately for comparison purposes. As the results show us, 1-grams produce better accuracy scores than 2-grams considering all the classifiers. There is a significant difference in accuracy scores of XgBoost and SVC-1 (with linear kernel) classifiers (approximately %12) for different n-gram settings. On average, classifiers performed 5%, 8%, 10% better in unigrams compared to bigrams for Dataset-10, Dataset-15, and Dataset-20, respectively. Let's continue with the table.

Table 4.1 Accuracy scores of the classifiers

| | **Algortihms** | **Dataset-10** | **Dataset-15** | **Dataset-20** |
|---|---|---|---|---|
| Unigrams | XgBoost | 0.968 | 0.956 | 0.946 |
| | SGD | 0.975 | 0.976 | 0.971 |
| | LR | 0.962 | 0.956 | 0.954 |
| | SVC-1 | 0.873 | 0.840 | 0.815 |
| | SVC-2 | 0.945 | 0.932 | 0.926 |
| | MLP | 0.915 | 0.846 | 0.710 |
| | NB | 0.832 | 0.775 | 0.746 |
| | Extra Trees | 0.936 | 0.904 | 0.871 |
| Bigrams | XgBoost | 0.831 | 0.855 | 0.822 |
| | SGD | 0.932 | 0.918 | 0.900 |
| | LR | 0.935 | 0.920 | 0.900 |
| | SVC-1 | 0.812 | 0.707 | 0.633 |
| | SVC-2 | 0.913 | 0.886 | 0.863 |
| | MLP | 0.896 | 0.765 | 0.607 |
| | NB | 0.793 | 0.712 | 0.668 |
| | Extra Trees | 0.875 | 0.820 | 0.768 |
| Transfer Learning | GPT-2 | 0.679 | 0.617 | 0.584 |
| | BERT | 0.986 | 0.853 | 0.839 |
| Average | | 0.873 | 0.861 | 0.819 |

We used GPT-2 and BERT as a transfer learning method. In fact, the part that took all our time was the stage of running the transfer learning algorithms, as it should be. Most of the articles in our dataset were long articles, so it made up a large dataset for us. The initial version of our dataset was equal to 510mb. Afterwards, we divided into 10, 15 and 20 authors and applied the necessary pre-processing, and we got a 147 MB file. Reducing the number of authors at this stage, and the preprocessing stage of the words reduced the file size.

Even this file size can be considered a high size when working with transfer learning algorithms, at least for our devices, we had crashes due to the ram memory being full during the continuous code execution phase. Let's take a deeper look together.

Tokenizer provides the preparation of mathematical values necessary for the algorithm. We provided the preparation of words using the tokenizer in Google BERT. You can see in the example. This is sample text in Figure 4.1.

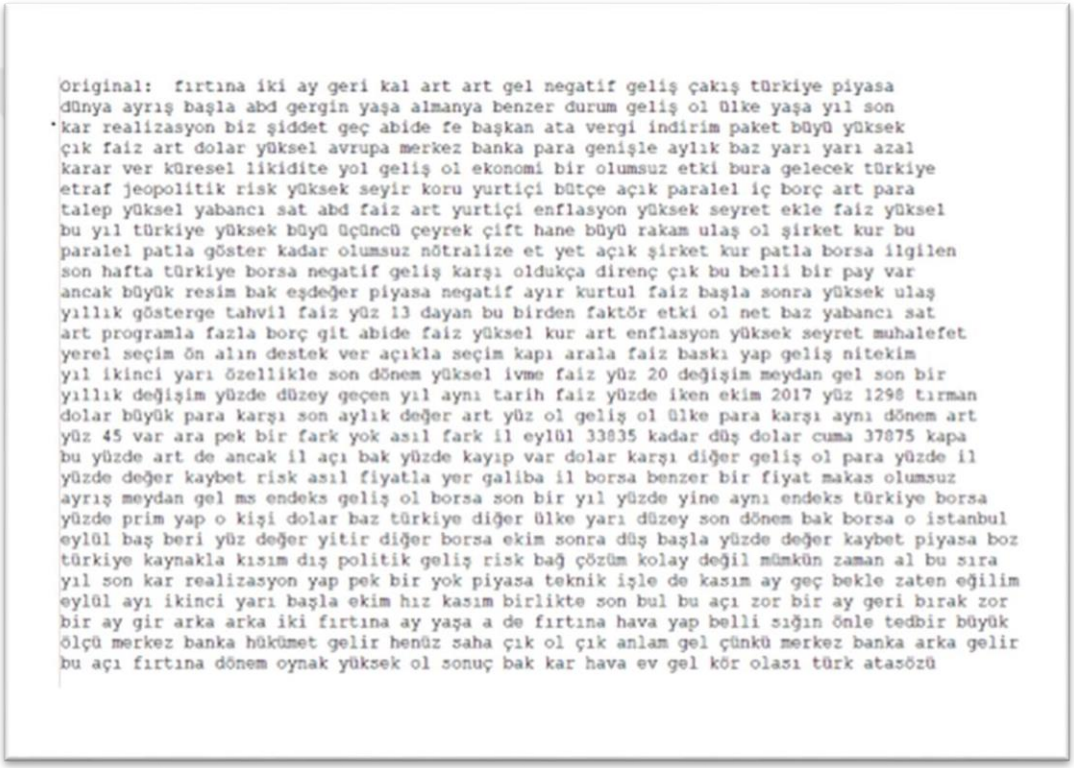


Figure 4.1 Sample text for algorithm

Each text is given to the algorithm and meaningful results are drawn by the algorithm because both machine learning algorithms and transfer learning algorithms cannot work with textual data. Therefore, it needs to be translated into numerical values that algorithms can understand. These operations are undertaken by a structure called tokenizer in transfer learning algorithms, we can see the code example in Figure 4.2.

```
input_ids = []
attention_masks = []

for text in training_texts:
    encoded_dict = tokenizer.encode_plus(
                        text,
                        add_special_tokens = True,
                        max_length = max_len,
                        pad_to_max_length = True,
                        return_attention_mask = True,
                        return_tensors = 'pt',
                   )

    input_ids.append(encoded_dict['input_ids'])
    attention_masks.append(encoded_dict['attention_mask'])

input_ids = torch.cat(input_ids, dim=0)
attention_masks = torch.cat(attention_masks, dim=0)
labels = torch.tensor(training_labels)

print('Original: ', training_texts[0])
print('Token IDs:', input_ids[0])
```

Figure 4.2 Tokenizer structure for transfer learning

```
Token IDs: tensor([     2,  11431,   2537,   2054,   3053,   2177,   2441,   2441,   2049,
        11537,  31009,  40770,   2076,  22918,   5380,  40395,   2613,   1017,
        68125,   2869,   8971,   4121,   4544,   3835,   2420,  31009,   1968,
        87717,   4121,   2124,   2039,   2029,   9710,   5100,   2251,  68090,
         3538,  61956,   7813,  34449,   4107,   4916,   7069,   5071, 122388,
        42064,  18101,   4986,   2441,   3612,  73923,   3038,   2581,   3730,
         2962,  40565,   1939,   4727,   2556,   5007,   5007,   5338,   2686,
         2053,  82360,  34977,   2336,  31009,   1968,   5758,   1947,   5147,
         6386,   4207,   3658,  22918,   5330,  33327,   4541,  42064,   6441,
         4318,  12245,   2155,  15186,   2052,  20314,   8395,   4413,   9271,
         2441,   2962,   3912,  73923,   3846,   2490,   2869,   4986,   2441,
        12245,   2155,   6487,  42064,   8390,  10321,   4986,  73923,   1964,
         2124,  22918,  42064, 122388,  93345,   2274,   9049,   5287,  59615,
        13543, 122388,   5742,  33204,   1968,  21876,   2083,   1964,   8395,
         3278,   1925,  30442,   2185,   5147,   4357,  72679,   5117,   2041,
         2385,  20314,  21876,   2083,   3278,   1925,   7365,   5780,   2039,
         3009,  22918,   7365,  11537,  31009,  40209,  40180,  84537,  18101,
         1964,   3979,   1947,   2943,   2130,   2591,  20086,   3941,   2182,
         2292,   1942,   4829,   5380,  11537,   4731,   4710,   4986,  68125,
         2244,  42064,  33204,   4038,  30442,   3039,  12377,   4986,  21663,
         3251,   3927,   1964,   5546,  45523,   6386,   1968,   3542,   2556,
         3846,   2490,   2441,  28450,   2554,   9271,   2669,  61956,   4986,
        73923,   2083,   2441,   6487,  42064,   8390,   7649,   4756,  75030,
         2121,   2618,   2871,   2053,  20314,   1925,  75030,   4143,  31288,
         1007,   4986,   5060,   1997,  31009,   9958,   2124,   3547,   5007,
        49138,   2039,  54387,  73923,  21933,   4986,  21663,   2115, 119646,
         5492,   2049,   2039,   1947,   4038, 119646, 104300,  26679,   1973,
        24727,   2124,   2573,   2388,   4986, 104300,   6456,   4114,   3721,
        21663,  28871,   1047,   9795,   3612,  20086,   2962,  40209,   2039,
         4727,  19680,   2441,  21663,   1968,  31009,   1968,  87717,   2962,
        40209,   2573,  54387,   2441,  21663,   5181,   2130,   2149,   3082,
         1947,   2431,   2351,   5141,   2431,   2068,  97876,  56354,  13716,
         2185,  21565,   3612,   5715,  58707,  15255,  18381,   1964, 104300,
         2441,   1961,   2591,   2068,   4378,   2182, 104300,   5962,   2130,
         3612,  40209,  17145,  31009,   1968,   2962, 104300,   2068, 104300,
```

Figure 4.3 Generated token ids

Generated token ids can seen in Figure 4.3. Each number represents a word. The creators of BERT have set a maximum of 512 tokens for each row, if we think from our dataset for each row equals the one article. The creators says if we increase the token size, the performance decreases in the experiments, and by placing such a limit, quality data entry is provided. Adjusting the data set according to these rules requires long efforts.

What if more than 512 words of data is given? Tokens of the first 512 words are created and given to the algorithm or it is possible to choose which part should be given to the algorithm. You can set 256 head + 256 tail or 512 tail only or 512 head only and also you can create subtexts classifier each of them and combine the results. All cases can be examined and the best results can be obtained. I would recommend to try option 512 head tokens, and only if this is not good for you, consider the other options.

Giving 512 tokens as a value depends on the developer. If your dataset is too large, the number of tokens can be reduced, it provides to us saving time and resource consumption.

I want to give additional information. It can increase by 512 tokens, which can be achieved by retraining the BERT system, but even on google super devices this process took a very long time, completing this training process on our own computer can be seen as a waste of time.

If we back to accuracy scores, BERT is one of the most used algorithms with the most language support among transfer learning methods. After BERT, the most used transfer learning algorithm is GPT. Although there are transfer learning algorithms that support Turkish, these algorithms are not included in this study because these algorithms use BERT's infrastructure.

For our 10-author dataset, we achieved the best performance with the BERT algorithm as can see in Table 4.2 and also you can see detailed performance scores in

Table 4.3, Table 4.4, and Table 4.5. However, as the number of authors increases, as we explained earlier, the number of tokens increases, and this increases the load for the BERT algorithm. Therefore, there is a big difference between the performance of the dataset with 10 authors and the performance of the dataset with 15 authors. As we can see in the two transfer learning algorithms we used in this study, there are special variables such as maximum tokens and vocabulary size and the values above are ignored. This makes it difficult for the algorithm to distinguish between authors. Because there may be words that the authors use a lot or there may be special words belonging to the branches of the authors. Due to the missing of these words, a significant decrease in performance was observed again. Let's take a look at performance score by author. Code examples can be seen in Figure 4.4.

Table 4.2 Bert performance scores

| Dataset | F-score | Recall | Precision |
|---|---|---|---|
| Author-10 | 0.9861 | 0.9859 | 0.9863 |
| Author-15 | 0.8539 | 0.8537 | 0.8551 |
| Author-20 | 0.8391 | 0.8389 | 0.8409 |

```
    for step, batch in enumerate(train_dataloader):
        if step % 10 == 0 and not step == 0:
            elapsed = format_time(time.time() - t0)
            print('Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format(step, len(train_dataloader), elapsed))

        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)

        model.zero_grad()
        output = model(b_input_ids,
                       token_type_ids=None,
                       attention_mask=b_input_mask,
                       labels=b_labels)
        loss = output['loss']
        logits = output['logits']
        total_train_loss += loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()

    avg_train_loss = total_train_loss / len(train_dataloader)
    training_time = format_time(time.time() - t0)

    print("Average training loss: {0:.2f}".format(avg_train_loss))
    print("Training epoch took: {:}".format(training_time))

    training_stats.append(
        {
            'epoch': epoch_i + 1,
            'Training Loss': avg_train_loss,
            'Training Time': training_time,
        }
    )

print("Training completed in {:} (h:mm:ss)".format(format_time(time.time()-total_t0)))
```

Figure 4.4 Code example of bert algorithm

Table 4.3 Performance comparisons by author on 10 author datasets

| Authors | Precision | Recall | F1-Score |
|---|---|---|---|
| Authors 1 | 0.9891 | 0.9981 | 0.9936 |
| Authors 2 | 0.9774 | 0.9825 | 0.9799 |
| Authors 3 | 0.9817 | 0.9942 | 0.9907 |
| Authors 4 | 0.9872 | 0.9942 | 0.9907 |
| Authors 5 | 0.9927 | 0.9837 | 0.9882 |
| Authors 6 | 0.9779 | 0.9881 | 0.9830 |
| Authors 7 | 0.9896 | 0.9812 | 0.9854 |
| Authors 8 | 0.9919 | 0.9935 | 0.9927 |
| Authors 9 | 0.9794 | 0.9837 | 0.9815 |
| Authors 10 | 0.9964 | 0.9775 | 0.9869 |

Table 4.4 Performance comparisons by author on 15 author datasets

| Authors | Precision | Recall | F1-Score |
|---|---|---|---|
| Authors 1 | 0.906367 | 0.908068 | 0.907216 |
| Authors 2 | 0.937729 | 0.934307 | 0.936015 |
| Authors 3 | 0.888889 | 0.832653 | 0.859852 |
| Authors 4 | 0.935421 | 0.886827 | 0.910476 |
| Authors 5 | 0.802083 | 0.806283 | 0.804178 |
| Authors 6 | 0.776490 | 0.851180 | 0.812121 |
| Authors 7 | 0.843407 | 0.878398 | 0.860547 |
| Authors 8 | 0.821732 | 0.872072 | 0.846154 |
| Authors 9 | 0.854815 | 0.856083 | 0.855448 |
| Authors 10 | 0.711864 | 0.739726 | 0.725528 |
| Authors 11 | 0.924632 | 0.856899 | 0.889478 |
| Authors 12 | 0.886282 | 0.919476 | 0.902574 |
| Authors 13 | 0.891374 | 0.904376 | 0.897828 |
| Authors 14 | 0.835182 | 0.780741 | 0.807044 |
| Authors 15 | 0.810376 | 0.779690 | 0.794737 |

Table 4.5 Performance comparisons by author on 20 author datasets

| Authors | Precision | Recall | F1-Score |
|---|---|---|---|
| Authors 1 | 0.903592 | 0.898496 | 0.901037 |
| Authors 2 | 0.931034 | 0.934426 | 0.932727 |
| Authors 3 | 0.884615 | 0.893661 | 0.889115 |
| Authors 4 | 0.914563 | 0.880374 | 0.897143 |
| Authors 5 | 0.743346 | 0.857456 | 0.796334 |
| Authors 6 | 0.932632 | 0.932632 | 0.932632 |
| Authors 7 | 0.780000 | 0.816754 | 0.797954 |
| Authors 8 | 0.829222 | 0.793103 | 0.810761 |
| Authors 9 | 0.811791 | 0.758475 | 0.784228 |
| Authors 10 | 0.866667 | 0.874106 | 0.870370 |
| Authors 11 | 0.924632 | 0.856899 | 0.889478 |
| Authors 12 | 0.886282 | 0.919476 | 0.902574 |
| Authors 13 | 0.891374 | 0.904376 | 0.897828 |
| Authors 14 | 0.858191 | 0.783482 | 0.819137 |
| Authors 15 | 0.931947 | 0.839864 | 0.883513 |
| Authors 16 | 0.870855 | 0.937970 | 0.903167 |
| Authors 17 | 0.881890 | 0.907618 | 0.894569 |
| Authors 18 | 0.761836 | 0.785503 | 0.773489 |
| Authors 19 | 0.738426 | 0.668763 | 0.701870 |
| Authors 20 | 0.808905 | 0.749141 | 0.777877 |

In this study, although language independent studies were tried, the transfer learning method was tried to be used on Turkish texts trained in English or a different language. However, the different structure of the language and the different words, an accuracy result could not be obtained and was not included in this study.

Let’s talk about GPT. As before we said GPT developed by OpenAI company. The company is an AI research and deployment company. They have a lot of different algorithms for artificial intelligence. Their last algorithm is GPT 3. GPT mean Generative Pre-Trained Transformer, it tries to produce content similar to people write.

They have also developed that chatgpt software, which has been mentioned frequently in the last months, and this software is a software that can show results like Google. Although it is shown as a software that can bring the end of google by some technology magazines, it gives clear answers in research assignments or question-answer questions.

Moreover, chatgpt, which can provide a conversational environment, also establishes meaningful connections from the previous spoken texts and offers a logical chat environment. In this respect, they can create the feeling of talking to a real person. When asked, it argues that it is not an artificial intelligence product. This application, which is still in beta stage, already shows that it can be used frequently when searching for information on the internet.

Before using this algorithm, the necessary environment had to be prepared. Since the hardware of our system was not sufficient during the execution of this algorithm, we developed on Google Colab Pro.

We installed torch 1.10.2, torchvision 0.11, torchaudio 0.10, sklearn, tqdm and transformers. If you are using higher or lower version of them, it will be uninstalled and that version will be downloaded and installed. After that stage 6767 MB file downloaded for our device.

As we mentioned before, the BERT algorithm was taking 512 token values. In GPT-2, although it is the same system like a BERT algorithm, It receives 1024 tokens with default settings and this number is not the maximum size. Even tough it is default settings, that is exactly twice bigger than the BERT algorithm. A GPT-2 algorithms can handle with 50257 different vocabulary size. Let’s take a look GPT-2 default config settings in Table 4.6. No changes were made in normal config settings. Training was performed in the settings you can see in this table. The vocabulary size could have been lowered to reduce the training time of algorithm, but it was not preferred because the number of words was too high in our dataset. Making changes to this variable will negatively affect performance.

Table 4.6 Default gpt-2 config settings

| | |
|---|---|
| Vocab_size | 50257 |
| N_positions | 1024 |
| N_emd | 768 |
| N_layer | 12 |
| Activation_function | Gelu_new |
| Embd_pdrop | 0.1 |
| Layer_norm_epsilon | 1e-05 |

Construct a GPT-2 tokenizer. Based on byte-level byte-pair-encoding. This tokenizer pays attention to the spaces between words, even at the beginning and end of words. Let's give two example in Figure 4.5 and in Figure 4.6.


Figure 4.5 Gpt-2 tokenizer example


Figure 4.6 Gpt-2 tokenizer example 2

If you pay attention to the examples, there is a space before hello in the second picture. This space caused the token value to change for the word hello. In both

examples, the dot is represented by token number 13. Even if it is run with this code more than once or if run is done on different devices, Same token numbers are given to the same words. Token numbers are predetermined for words.

BERT was work only 4 epochs, GPT-2 was worked 5 epochs. Transfer learning algorithms do not need to be run 50 100 epoch times like machine learning algorithms. The time taken for one epoch 08 minutes 55 seconds for the BERT algorithm and it took 13 minutes 48 seconds for the GPT 2 algorithm for one epoch. Dimensionality of the embeddings and hidden states is default 768 for both algorithms and number of hidden layers is 12 for both algorithms. Vocabulary size for BERT default 30522. If you will work with more than 30522 words, it needs to be increase that integer number.

Let's look at GPT-2 Performance scores in 10 authors in Table 4.7. You can check the performance scores in 15 authors in Table 4.8 and performance scores in 20 authors in Table 4.9.

Table 4.7 Gpt-2 performance in 10 authors

| **Epochs** | **Train Loss** | **Train Accuracy** | **Validation Loss** | **Validation Acc.** |
|---|---|---|---|---|
| 1 | 0.718 | 0.509 | 0.522 | 0.643 |
| 2 | 0.282 | 0.810 | 0.516 | 0.664 |
| 3 | 0.059 | 0.963 | 0.658 | 0.659 |
| 4 | 0.032 | 0.980 | 0.703 | 0.677 |
| 5 | 0.025 | 0.983 | 0.804 | 0.651 |

Table 4.8 Gpt-2 performance in 15 authors

| **Epochs** | **Train Loss** | **Train Accuracy** | **Validation Loss** | **Validation Acc.** |
|---|---|---|---|---|
| 1 | 0.814 | 0.486 | 0.634 | 0.604 |
| 2 | 0.337 | 0.788 | 0.644 | 0.629 |
| 3 | 0.069 | 0.957 | 0.858 | 0.611 |
| 4 | 0.036 | 0.977 | 0.879 | 0.622 |
| 5 | 0.027 | 0.983 | 0.994 | 0.617 |

Table 4.9 Gpt-2 performance in 20 authors

| Epochs | Train Loss | Train Accuracy | Validation Loss | Validation Acc. |
|---|---|---|---|---|
| 1 | 0.920 | 0.447 | 0.727 | 0.561 |
| 2 | 0.383 | 0.767 | 0.707 | 0.597 |
| 3 | 0.073 | 0.956 | 0.947 | 0.577 |
| 4 | 0.040 | 0.975 | 1.046 | 0.581 |
| 5 | 0.030 | 0.981 | 1.124 | 0.576 |

Let's look at the tables above. For want of a better term, loss really monitors the inverse-confidence of the prediction. The model may be too sophisticated for the data, or it may have been trained for a lengthy time, as a noteworthy cause of this occurrence. To get a better score, the number of words in the articles can be reduced, so we can give data in size that the algorithm can deal with. While the BERT algorithm works well enough our data, it has become complex and large for the GPT algorithm.

# CHAPTER FIVE
# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

Recently, with the development of artificial intelligence, the capabilities of algorithms have increased. From the perspective of our own work, text summarization, text generation and studies in the field of text mining capabilities have increased considerably. Artificial intelligence now produces good results in the prediction of the next words. This is also the system that Google currently uses in its search engine. sorts what you can potentially type after typing a word in the search bar, this is provided by the Google BERT. Even if we do not realize it, transfer learning applications appear on our computers and phones that we use almost every day in our daily lives.

In this study, A large amount of news articles has been collected and various text cleaning and pre-processing operations have been applied on it. Three different sizes of the author datasets have been created and unigram and bigram features have been investigated on these datasets. We also used TF-IDF to expose the author's specific n-gram features. We set up the maximum features parameter of TFidfVectorizer as 3000 to build a vocabulary that only considers the top 3000 features ordered by term frequency across the particular datasets. Some important machine learning and ensemble learning algorithms and also transfer learning methods that have been applied to different fields and have shown successful performances were trained on the author datasets. The GridSearchCV exhaustive search technique has been employed in the hyperparameter selection process. For all combination of values in the specified range, the network is trained and selected with the best hyperparameter for best accuracy rate.

We compared machine learning algorithms and transfer learning methods, so we saw the effectiveness of transfer learning algorithms, which is the main topic of our study. Transfer learning algorithms are both large in size and need powerful systems

during the execution of these algorithms, so it is not one of the first methods that comes to mind for people who have dealt with text mining and have worked on these issues. Although there are many alternatives as a transfer learning method, most algorithms use BERT infrastructure. At the same time, since the articles in the dataset we work with are written in Turkish, most algorithms do not have Turkish language support.

Since millions of articles were used during the training of transfer learning algorithms, we did not have a chance to make an algorithm available for Turkish. For this reason, we have carried out our work with two transfer learning algorithms, which are the most used and have Turkish support. Since the number of articles in our dataset and the number of words in the articles are huge, the training time of the algorithms takes quite a long time. which means that it will take a lot of time to train algorithms in 10 epochs or more.

As far as we have observed, operating more than 5 epochs, we cannot observe significant increases in performance. When we with our dataset of 10 authors, we achieved the best performance score among the algorithms. however, when we increased the number of authors, we could not perform the operation to create tokens for all words even on google colab pro, because although google colab allows us to work in a computer environment with 80 gb ram, the algorithm said that we need to have 180 gb ram. Since we could not provide such an environment, we had to reduce the number of tokens. This may have resulted in low performance for 15 and 20 authors. Although transfer learning algorithms require a lot of resource when working with large data sets, they can achieve performances that machine learning or deep learning algorithms cannot reach in few data datasets. In datasets with few data, noticeably larger differences emerge.

It has been found that the most successful method for 10 authors dataset is BERT transfer learning method according to performance evaluation metrics. It is followed by the SGD, XgBoost and LR methods with almost similar performance. In our other datasets with 15 and 20 authors, the SGD algorithm achieved the best performance.

This shows that it can actually achieve sufficient levels of performance in machine learning algorithms.

### 5.2 Future Works

As a future works, by assigning a fixed value to each author for the number of articles in our dataset, we can prepare a data set that is easier to work with and increase the efficiency we will get from transfer learning algorithms. As we mentioned before BERT and GPT algorithms had different structures. The number of articles and the number of words in the articles was too high. In order to get better performance than the GPT algorithm for the Turkish language, the dataset size can be reduced. In addition, we used TF-IDF vectorizer in this study. Apart from this method, there are methods such as fasttext, glove, word2vec that we can use. By integrating them, it can be tried to achieve better performance on the data.

## REFERENCES


Akın, A. A., & Akın, M. D. (2007). Zemberek, an open source NLP framework for Turkic languages. Structure, 10(2007), 1-5.

Alhuqail, N. K. (2021). Author identification based on nlp. European Journal of Computer Science and Information Technology, 9(1), 1-26.

Alonso-Fernandez, F., Belvisi, N. M. S., Hernandez-Diaz, K., Muhammad, N., & Bigun, J. (2021). Writer identification using microblogging texts for social media forensics. IEEE Transactions on Biometrics, Behavior, and Identity Science, 3(3), 405-426.

Atar, M. S., Esen, E., & Arabaci, M. A. (2018, May). Supervised author recognition with aggregated word embeddings. In 2018 26th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

Bandara, U., & Wijayarathna, G. (2013). Source code author identification with unsupervised feature learning. Pattern Recognition Letters, 34(3), 330-334.

Barlas, G., & Stamatatos, E. (2020, June). Cross-domain authorship attribution using pre-trained language models. In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 255-266). Springer, Cham.

Damiran, Z., & Altangerel, K. (2014, July). Author Identification-An Experiment Based on Mongolian Literature Using Decision Trees. In 2014 7th International Conference on Ubi-Media Computing and Workshops (pp. 186-189). IEEE.

De Vel, O., Anderson, A., Corney, M., & Mohay, G. (2001). Mining e-mail content for author identification forensics. ACM Sigmod Record, 30(4), 55-64.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Digamberrao, K. S., & Prasad, R. S. (2018). Author identification using sequential minimal optimization with rule-based decision tree on Indian literature in Marathi. Procedia computer science, 132, 1086-1101.

Diri, B., & Amasyalı, M. F. (2003, June). Automatic author detection for Turkish texts. In Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP) (pp. 138-141).

Fabien, M., Villatoro-Tello, E., Motlicek, P., & Parida, S. (2020, December). BertAA: BERT fine-tuning for Authorship Attribution. In Proceedings of the 17th International Conference on Natural Language Processing (ICON) (pp. 127-137).

Fávero, E. M. D. B., & Casanova, D. (2021). BERT_SE: A Pre-trained Language Representation Model for Software Engineering. arXiv preprint arXiv:2112.00699.

Fedotova, A., Romanov, A., Kurtukova, A., & Shelupanov, A. (2021). Authorship Attribution of Social Media and Literary Russian-Language Texts Using Machine Learning Methods and Feature Selection. Future Internet, 14(1), 4.

Fourkioti, O., Symeonidis, S., & Arampatzis, A. (2019). Language models and fusion for authorship attribution. Information Processing & Management, 56(6), 102061.

Karaman, B. I., Dalkılıç, F., & Eker, E. E. (2020). Author Recognition In Modern Turkish For Forensic Linguistic Cases Using Machine Learning. 17th National Forensic Science Congress.

Kırmacı, B., & Oğul, H. (2015, September). Evaluating text features for lyrics-based songwriter prediction. In 2015 IEEE 19th International Conference on Intelligent Engineering Systems (INES) (pp. 405-409). IEEE.

Kuyumcu, B., Buluz, B., & Kömeçoğlu, Y. (2019, April). Author Identification in Turkish Documents with Ridge Regression Analysis. In 2019 27th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

Mohsen, A. M., El-Makky, N. M., & Ghanem, N. (2016, December). Author identification using deep learning. In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 898-903). IEEE.

Okuno, S., Asai, H., & Yamana, H. (2014, October). A challenge of authorship identification for ten-thousand-scale microblog users. In 2014 IEEE international conference on big data (big data) (pp. 52-54). IEEE.

Oliveira Jr, W., Justino, E., & Oliveira, L. S. (2013). Comparing compression models for authorship attribution. Forensic science international, 228(1-3), 100-104.

Otoom, A. F., Abdullah, E. E., Jaafer, S., Hamdallh, A., & Amer, D. (2014, April). Towards author identification of Arabic text articles. In 2014 5th International conference on information and communication systems (ICICS) (pp. 1-4). IEEE.

Örücü, F., & Dalkılıç, G. (2010). Author Identification Using N-grams and SVM. 17th National Forensic Science Congress, 1, 3-5.

Paci, H., Kajo, E., Trandafili, E., Tafa, I., & Salillari, D. (2011, September). Author identification in Albanian language. In 2011 14th International Conference on Network-Based Information Systems (pp. 425-430). IEEE.

Pandian, A., Ramalingam, V. V., & Preet, R. V. (2016). Authorship identification for Tamil classical poem (Mukkoodar Pallu) using C4. 5 algorithm. Indian Journal of Science and Technology, 9(46).

Polignano, M., Gemmis, M. D., & Semeraro, G. (2020, July). Contextualized BERT sentence embeddings for author profiling: The cost of performances. In International Conference on Computational Science and Its Applications (pp. 135-149). Springer, Cham.

Ramezani, R. (2021). A language-independent authorship attribution approach for author identification of text documents. Expert Systems with Applications, 180, 115139.

Ramezani, R., Sheydaei, N., & Kahani, M. (2013). Evaluating the effects of textual features on authorship attribution accuracy. In ICCKE 2013 (pp. 108-113). IEEE.

Romanov, A., Kurtukova, A., Shelupanov, A., Fedotova, A., & Goncharov, V. (2020). Authorship identification of a russian-language text using support vector machine and deep neural networks. Future Internet, 13(1), 3.

Saedi, C., & Dras, M. (2021). Siamese networks for large-scale author identification. Computer Speech & Language, 70, 101241.

Sage, M., Cruciata, P., Abdo, R., Cheung, J. C. K., & Zhao, Y. F. (2020). Investigating the Influence of Selected Linguistic Features on Authorship Attribution using German News Articles. In SwissText/KONVENS.

Sari, Y., Stevenson, M., & Vlachos, A. (2018, August). Topic or style? exploring the most useful features for authorship attribution. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 343-353).

Stamatatos, E. (2009). A survey of modern authorship attribution methods. Journal of the American Society for information Science and Technology, 60(3), 538-556.