Journal of Theoretical and Applied Information Technology
31st October 2020. Vol.98. No 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AUTHORSHIP VERIFICATION OF TWEETS CROSS TOPICS USING WEIGHTED WORD VECTORS SIMILARITY

[1]SHAIMAA AYMAN, [2]MOHAMED EISA,[2]FIFI FAROUK

[1]Department of Computer Science, Faculty of Computers and Information, Zagazig University, Sharqiyah, Egypt.

[2]Department of Technology and Information System, Port Said University, Egypt.

E-mail: [1]shayman@zu.edu.eg, [2]mmmeisa@yahoo.com, [2]bondoka7000@himc.psu.edu.eg

ABSTRACT

Authorship Verification (AV) is one of the interesting topics that had developed rapidly and distinctly since the middle of the 19th century. With the social media era, there is always a problem in determining whether a given tweet, post, or comment was written by a certain user or not. We are proposing a new approach to verify if a tweet belongs to a claimed user. Our proposed method utilizes the benefits of one-shot learning. It is based on vectors similarity which depends on Term Frequency–Inverse Document Frequency (TF-IDF) and word embedding for better verification accuracy. After comparisons, our proposed approach outperforms existing methods in the case of cross topics.

Keywords: *Authorship Verification, Vectors Similarity, TF-IDF, Word Embedding, Cross Topics.*

1. INTRODUCTION

Digital information networks and social media platforms are developing and spreading, leads to a growing necessity for reliable applications for plagiarism detection [1] and author verification [2]. Authorship verification is the application of linguistic learning in the context of traditional stylometry, also known as forensic text comparison, two or more text documents are compared with stylometric of ascertaining authors to determine if the documents were written by the same author or not. The analysis of these documents depends on traditional linguistic categories. Which include spelling/grammar, dialects, and stylistic behaviors. These linguistic features have become highly instrumental in authorship verification and analysis [3]. The problem we face can be considered a one-classification problem [4]; this means that a text is classified by its belonging or the given class (related to the alleged author) or (related to a fake author). For example, the Federal Criminal Police of Germany (Bundeskriminalamt) deals with ways of comparing the text to implicitly identify suspects and confirm or deny charges against the defendant in a criminal case [5]. Authorship verification not only gathers tangible evidence in criminal investigations but also discovers deceitful intentions and fake news in social media.

Computer scientists and engineers have begun to reduce the threat to identity theft and verify the authors' information. The objective of authorship verification is to discover whether two different documents were written by the same author or not [6]. Automated text categorization plays an important part in this process where topic, sentiment, and style of documents can be used as discriminating factors.

Authorship analysis methods have depended on the extraction of stylometric features [7]. Stylometry is the behavioral feature of the author, which displays during his writing style, then it can be extracted and used to check the identity of the author of the online text. Represent and extract these features are the main problems in the text classification, many studies have been conducted in this field in recent years [8], [9]. But the disadvantage of these features is that their reliability is reduced in short text and cross topics. Therefore, social media texts remain challenging, and writing is not obliged to write grammar and spelling. So, we use short text for Term Frequency–Inverse Document Frequency (TF-IDF) vectorizer function from the scikit-learn library and word embeddings to extract features, which show outperform the stylometric features in social media texts.

There are some limitations we have encountered, first we deal with short Twitter texts with a

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

maximum block size of 140 characters per tweet. The second limitation in the type of data used, when the content of the text or message expresses the topic that a person is trying to talk about. The vocabulary used in the text determines the displayed topic. Thus, when the vocabulary changes, the topic also changes. Therefore, we are working on developing authorship verification which focuses on placing a weighted word vectors similarity in cross topics. These challenges were mainly affecting the performance of the system.

So, our objective is explicitly to solve the verification problem in cross topics which find the relation between the tweet's vectors and the user vector, based on the weight of each word depending on TF-IDF and Word2vec which based on Word Embeddings strategy to learn the best word representation and extract the text close to the original text in small dimensions. The proposed implementation aims at enhancing performance by combining the advantages of TF-IDF and Word embedding to distinguish between distinctive words and to distinguish between subjects by a weight vector.

The TF-IDF vectorizer [10] is used to convert a collection of raw documents into a matrix of TF-IDF features. TF-IDF is a numerical statistic that is intended to reflect on how important a word in a document. Word Embedding [11, 12] is a class of approaches to representing words and text using a dense vector representation and solve the problem of shortage data, which can retain the important syntax and semantics in the text and extract the meaning of this text which depends on a Word2Vec pre-trained word vector. That model must be mindful of the contextual similarity of words. In another sense, we use it to construct a low-dimensional vector representation from the text corpus and to enhance the contextual similarity of words.

This paper is organized as a background of recent works for authorship verification, shown in section 2. Section 3 displays our proposed approach. Section 4 shows the experiments that have been applied to our approach and the results of the final work. Finally, section 5 displays a conclusion and future work of this study.

2. RELATED WORK

Authorship verification is an emerging issue in authorship analysis, and it has been studied by many researchers who use different algorithms for machine learning and deal with a wide range of features, methods, and corpora. It has been implemented in several ways to the features and methods suggested before Koppel [13], which evaluate authorship verification by a method called "unmasking" using a collection of 21 English books in19th century written by 10 different authors with a variety of genres. The corpus is constructed 189 distinct different author pairs and 13 distinct same author pairs. they tried to determine the discrepancy between the suspect's sample document and that of other users (imposters). This approach can provide reliable results only for documents that are at least 500 words long, which is not realistic in the case of online verification. Chen and Hao [14] used 150 stylometric features for applying authorship similarity detection from e-mail messages which using 40 authors of the Enron dataset. The number and length of emails have impacted the final performance for several cases. The best result achieved when used SVM and decision tree as basic methods with the increasing length of e-mails, and the performance of PCA and K-means clustering outweighed in this research for all cases. There are some limitations in the length of e-mails as they erased messages that are less than 30 words length because they lack information capable of distinguishing the author. Brocardo [15] studied the effect of the possibility of using stylometry for authorship verification for short online messages. Based on the combination of supervised learning and n-gram analysis. They used the Enron emails dataset including 500 characters of block size for 87 authors. They used stylometric techniques through linguistic analysis and writing styles. They evaluated the performance of their approach through a 10-fold validation test. There are some limitations in their model used one type of features and not good also to handle short message content 10 to 50 characters like Twitter. In an attempt by Ammar Adil Abdulrazzaq [16] to assess the accuracy of the impact by applying the text of John Burroughs-Delta the Arabic way of revealing stylistic authorship to predict the appropriate author through his writing style depending on the frequency word as the best attribute. Their model focused on the Arabic literature and worked on the analysis based on word redundancy as a feature in Arabic books as a frequent, pair and trio-of-words and test the results obtained using a stylometric authorship attribution. This method does not deal with short messages their datasets that used 10 books which 6 books for the author which 5 of them for making learning map and 1 for testing, and 4 books for the other authors to test and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

compare. A dedicated shared task series at PAN 2015 CLEF datasets by (Maitra [17], NE. Benzebouchi, N.Azizi [18],). Maitra [17] used the Random Forest classifier for automatic software authorship verification to select important features from 17 kinds of features such as punctuation, vocabulary, length of sentence, N-gram, and POS. The best result has appeared in the Dutch language. Their performance of English and Spanish language declined due to the variable number and size of the known documents, and also appear cross-genre and cross-topic texts which reduce the performance. While that Benzebouchi, N. Azizi's [18] authorship verification system depending on the merger of many classifiers, which include Convolutional Neural Network (CNN), (Recurrent-CNN) and Support Vector Machines (SVM), and using word2vec for word embedding. Their model can learn meaningful texts without craftsmanship. The good performance appeared in the Skip-Gram technique than the CBOW technique. This experiment deals with 100 authors; containing 100 known documents and 100 unknown documents written in English only.

After many experiments and after various studies for the previous works we reached our proposed approach which verifies short text (tweets) in a cross topic, which used the vector of TF-IDF with the vector of word embedding to achieve the best performance using our own Twitter dataset.

We got inspired by our approach by the one-shot learning strategy using the word embedding technique [19], which explained the Long Short-Term Memory (LSTM) and its ability to deal with the pretrained word-embeddings and worked in modeling complex semantics. They took a pre-trained word-vectors as the LSTM inputs. Another research had learned an embedding into a Euclidean space for face verification. It didn't require more complex processing to handle their model [20].

3. PROPOSED APPROACH

The main idea of our approach is to have the ability to identify if a new user is similar to any user that we have. In our case of authorship verification, we use a somehow similar idea to predict if the tweet belongs to the same user or not; based on a trained model to predict (based on vectors similarity) if the tweet indeed belongs to this user or not. We proposed an approach Authorship Verification applying TF-IDF with word2vec (AV with TF-IDF & Word2vec). AV with TF-IDF & Word2vec approach consists of four main phases; to build that, as shown in figure 1. The first phase is called a dataset, it consists of three steps: prepare the Twitter corpus, data normalization, and cleaned tweets. The second phase is a vectorization which consisting of two steps: TF-IDF vectorization and amplified TF-IDF matrix. The third phase is a representation in which we apply Word2Vec representation on our data. The fourth phase is called multiplication in which; we apply a machine learning classifier.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

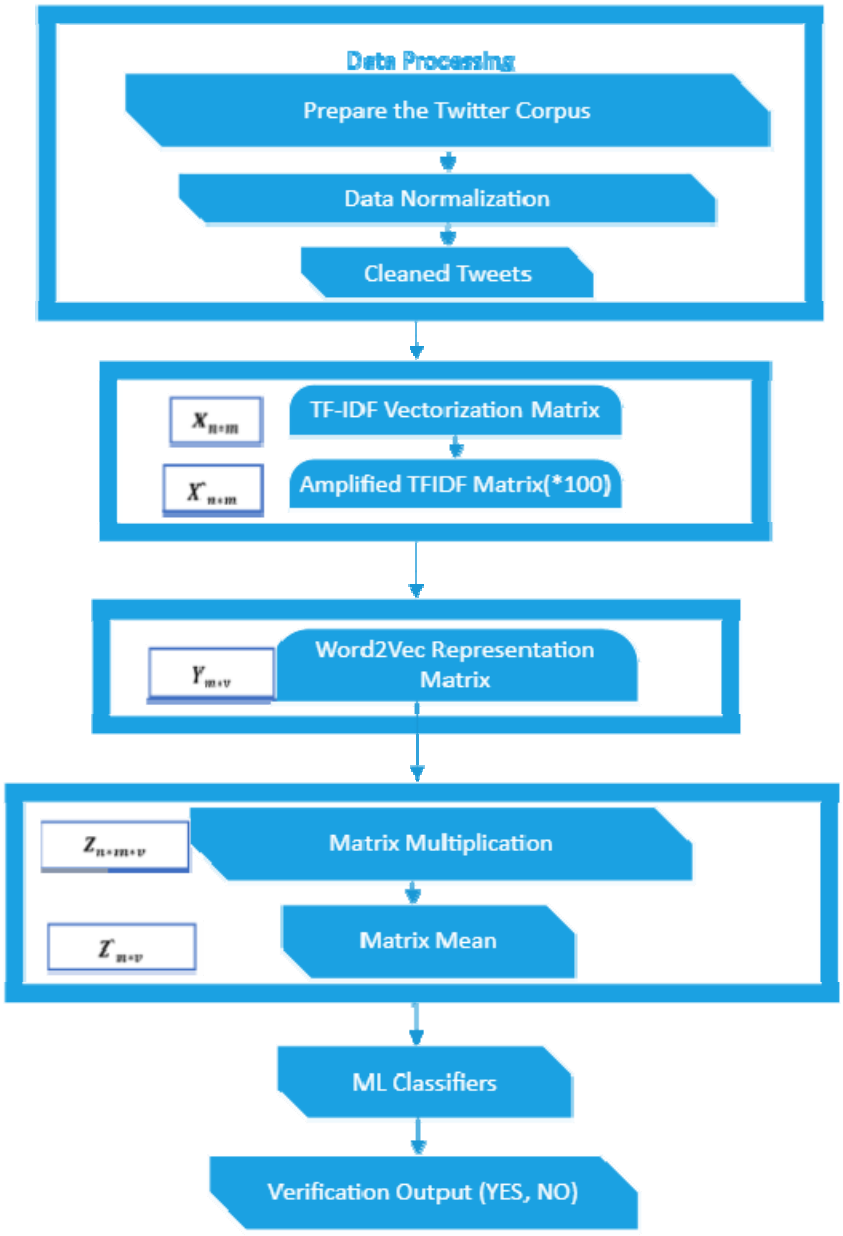

Figure 1 AV with TF-IDF & Word2vec Approach.

3.1. Phase One: Dataset

This phase is consisting of three steps:

- Collecting the Twitter corpus: This step uses to collect tweets manually from the Twitter dataset to apply as input to the proposed approach.
- Data normalization: we remove (all accent marks, extra white spaces, and all stop words) and replaces (emojis, emails, punctuation, symbols, and URL).
- Cleaned tweets: the data is ready that can be used to obtain our input.

3.2. Phase Two: TF-IDF Vectorization

TF-IDF[21] is a popular numerical statistic for information retrieval that measures the number of times a word occurs in a document, and through the whole corpus. We use a TF-IDF vector representation for 1500 words after preprocessing. Only the train set is used to compute the inverse document frequency values. To produce a TF-IDF

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

vectorizer use Scikit-learn which converts a collection of raw documents to a matrix of TF-IDF features [10], [22]. Equivalent to CountVectorizer followed by TfidfTransformer. By using CountVectorizer which convert a collection of text documents to a matrix of token counts and TF-IDF Transformer that converts a number matrix to a normal TF or TF-IDF representation, which calculate TF that is referred to the Term Document Frequency or some times that words appear in the document, and the IDF which refer to the Inverse Frequency of each word and place it with a weight depending on its importance in the context of speech. Calculate $TF_{i,j}$ (number of occurrences of i in j) in Eq. (1)

$$TF_{i,j} = \sum_{k} n_{i,j} \quad (1)$$

The IDF of a word is the measurement of how important that term is in the whole corpus shown in Eq. (2), where df_i (number of documents containing i) and N (total number of documents). Eq (3) we can obtain the weight of each word, $W_{i,j}$, as shown in Eq(3) TF-IDF.

$$IDF(W) = \log\left(\frac{N}{df_t}\right) \quad (2)$$

$$W_{t,j} = TF_{i,j} * \log\left(\frac{N}{df_t}\right) \quad (3)$$

So, Tf-Idf contains information on the more important word and less important one by its weight.

Amplified TF-IDF

We got Matrix from TF-IDF $X_{n*m.}$ which found the range values of TF-IDF are very small by default, we see all most of the weight of words has a small value. So, made the mathematical operation to make important words shine with boosted the TF-IDF matrix by a factor and multiply it by 100. Now we see that some very important words have emerged, and there are still some words that have a small value.

3.3. Phase Three: Word Embedding (Word2Vec) Representation

We use word2vec in word embedding strategy to give words high value to be able to distinguish between the words of each author. Create a set of vectors containing the count of word occurrence.

We can explain a word embedding as a technique of language modeling used to map words to real-number vectors. In vector space, it represents words or phrases of several dimensions. It can be created with different methods, such as probabilistic models, neural networks, matrix co-occurrence, etc. There are some applications of Word Embedding like sentiment analysis, information retrieval, speech recognition, and answering questions. The main idea of word embedding is that words occur in similar contexts, in vector space tend to be closer to each other. Word2Vec consists of the models for word embedding generation. These models are two-layers neural networks with one layer of input, one hidden layer, and one layer of output. Modules required for generating word vectors in Python are Natural Language Toolkit (Nltk) and genism libraries, which access to Word2Vec and other word embedding algorithms for training, and it allows pre-trained word embeddings that you can download from the internet to be loaded. Another library is designed to have fast performance, and with word embedding models built-in, it's perfect for an easy and quick start is called Spacy. It is a great way to represent a word and its meanings, we can observe the context of each tweet by simply get the vector of each word and get the tweet overall vector, for each word we need a vector of 300-word shape. We need to multiple each word by its vector to get the Word2Vec, the Spacy model is used for sentence boundary detection and tokenization which can be found here [11], [23].

Word2Vec makes use of two architectures:

- Skip Gram: It predicts the surrounding context words given the current word in a given framework[24].
- CBOW (continuous bag of words)[25]: This model predicts the current word of context words in a given framework.

The input layer consists of context words. The hidden layer consists of the number of dimensions in which we want the current word in the output layer to be represented. The output layer consists of the current word.

3.4. Phase Four: Multiply Amplified TF-IDF with Word2Vec

We can see that the whole word values are still relatively small. So, we are multiplying each word by its corresponding amplified TF-IDF, in this way we observe a greater weight vector for more important words. large matrix will be gotten with Z_{n*m*v}, where n refers to the number of observations, m refers to the number of words in corpus and v refers to the vector length for pre-trained word2vec. So, should be reduced the number of floats from 64 bits to 16 bits. Take the mean of this matrix to reduce the tweet to 2-dimensions matrix instead of 3-dimensions which transform each word into a vector. We used function to reduce the text matrix into one matrix

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

with a vector for each word, where we combine all the vectors from every tweet in the approach chosen in the approach parameter.

The final matrix will be got Z_{n*w}. Applying this way will be ending up having a vector for each word.

4. EXPERIMENTAL STUDY

We are ready now to show the experimental study for our proposed approach, we will be used two different data sets; the first one is our social Twitter dataset convert with the second one is the data of health news Twitter [26]. Which stored in the UCI machine learning repository to find the best results. To clarify the operations that occur in the experimental study in more detail; we can simply divide it into six stages, from section 4.1 to section 4.6.

In section (4.1) we show the experimental setup. Section (4.2) performs processing and operations on the data. Section (4.3) presents features and how to extract them. Section (4.4) present the training and classification processes. Section (4.5) show the evaluation criteria which measure the performance of the system. Section (4.6) provides results and discussion.

4.1. Experimental Set-Up Stage

The proposed approach was implemented with python3.7 using anaconda3 prompt. The software was Microsoft Windows 10. The hardware has Intel(R) Core (TM) i7-6500U CPU @ 2.50HZ 2.60HZ with 16G memory.

4.2. Data Processing Stage

This stage contains to process of collecting and normalizing the dataset.

4.2.1. Collected data

Twitter is a microblogging feature that lets authors post "tweets". Each tweet is limited to 140 characters and often shares opinions on various topics. Most programs written to access Twitter data provide a library that acts as an interpreter around the Twitter search and streaming API and is therefore constrained by API restrictions. One of these limitations is that you can only be sent 180 requests every 15 minutes with a maximum number of 100 tweets per request. The biggest drawback of the search API is that you've only been able to access the tweets written in the last seven days. This is the major impasse for anybody looking for old data to create any model.

So, we manually scraped data from Twitter. By using a username and user handler for each user to obtain Twitter corpora. We create our dataset of author classified tweets, which using a Twitter client application that randomly collects public statuses using Twitter Scraper. A python-based twitter corpus collected by a tool called taspinar (Twitter Scraper) [27]. With Twitter Scraper, no limitation existing in the API. We excluded people which their team typing to them on Twitter or many persons which people may write about them; since we achieve the credibility of our collected data. We tried to collect as much as possible the real public persons' tweets with more than topics like sports, politics, media, music, etc. It is natural for people to speak on different subjects. otherwise, the benefit will be very limited. This will be added strength to the approach. Because the system is learning more different words, and it is more distinguished between them.

4.2.2. Normalization data

The data has multiple users that represent multiple topics, we chose these users based on the number of tweets, as they have the most tweets in their domain. We used 30 users with 1000 tweets per user from different fields.

We made some operations to normalize our dataset; we used a module called CUCCO for text normalization and preprocessing [28], which helped us to remove accent marks, stop words, extra white spaces and replace punctuation, URL, emojis, replace emails, replace symbols). Also used for analysis data library called Pandas [29]. The final form of data after processing like as Table 1.

Table 1 Final Form Data.

Index of tweet	Text	user
24	yes reverse https twitter comhayson tweets status num	Kristen Bell
657	toutes mes felicitations au pm abeshinzo pour ...	Justin Trudeau
467	Turn bull government plan fairer share gst leav...	Scott Morrison
791	num york times reported good helps give recei...	Bill Gates
96	num lilydpenha happy birthday love	Amitabh Bachchan
901	labor plans grow economy plans suffocate highe...	Scott Morrison
1041	court skyscraperhttpstwittercomchefcourt num s...	Dwayne Johnson
1327	num monthshttpstwittercomrudrasrk num status num	Shah Rukh Khan
908	congrats orlando ortega silver medal rio num	Rafa Nadal
1139	twitter deal sony num num lens worn works per...	Jonathan Morrison

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.3. Feature Extraction

Now we can propose a feature extraction based on Word Embeddings and TF-IDF as the following steps:

- Train word embeddings by using word2vec which supplied by the Spacy package [23].
- To get the vector representation for each word we can use Spacy. This is an open-source library for Natural Language Processing (NLP) written in Python which used for parsing, tagging, and entity recognition. Spacy has been prepared and executed to give us a balance of accuracy, speed, and size. Embedding strategy with subword features is used to support huge vocabularies in small tables.
- Used the large English model on Spacy [30] which consists of over 1 million unique vectors with vocabulary, syntax, entities, and written text like blogs, news, and comments. Its vector representation consists of 300 dimensions.
- Then train a TFIDFVectorizer [22] to get the TF-IDF vector for each word in the whole corpus. Which able to distinguish the authors by their words.
- The TF-IDF technique calculates the inverse document frequency for each word. By doing it; can weight words based on its importance in the context. Now each word has a weight, and we have a vector for each word. We used only the most important words to have a higher impact on the user vector.

4.4. Training and Classification Stage

This stage contains training and classification processes.

4.4.1. Training process

Our data contains about 30 users, 50 words per user, and 300-word shape per word which has been much training for the best performance. Now, we show the steps that have been applied for training to get the best performance.

- The collected data were grouped according to the users. Then, it has been normalized using the CUCCO model, the data is divided into a train and test data. The training data was conducted and then transformed the data to vectors.
- Matrix X with a user and tweet was obtained, and Y had a true or false indicating whether the tweet belonged to that user. Every user's tweet was compared to all the tweets of other users, and therefore we could get very large data to train our approach to that, although the data was balanced, we took all tweets and user tweets from other users equally.
- The train and test data were saved on disk then, loaded them from disks and return (x_train, y_train), and (x_test, y_test). Then an organized x, y matrix was got. In the training, we have provided the data in Table 2. Where W in this table refers to a vector weight for each word, and n refers to the numbers of words for each user. The tweet vector was built using the Spacy tool. The tweets we used in the testing data was different from the tweets used to build the vector of the training.

Table 2 Weighted Vector of The User.

Weighted tweet vector	belong to
$[W_{01}, W_{02}, \ldots. W_{0n}]$	user_0
$[W_{11}, W_{12}, \ldots. W_{1n}]$	user_1
$[W_{21}, W_{22}, \ldots. W_{2n}]$	user_2
$[W_{31}, W_{32}, \ldots. W_{3n}]$	user_3

- After completing all the operations on the data, we want to reduce the matrix of three-dimensional tweets to two-dimensional by reducing the text matrix to one matrix, which combines all vectors from each tweet in the chosen method in the parameters that convert one tweet [1500 * 300] into one vector [300].
- The trainer function has re-trained on a text matrix to build the words vectors matrix, then saves it.
- As we know that the Spacy vector size is 300 dimensions, so the vector for each word is obtained and saved. Then convert the text matrix to the corresponding word2vec matrix, where each text is replaced by a matrix with shape (N, 300) (N: the number of words in the TF-IDF model, 300: word2vec length).
- The text was transformed to TF-IDF, multiplied the idfs by factor. Then multiplied each word by its vector. The matrix was obtained with train size 30000 tweets for all uses, 300 vectors for each word, and 1500 words for tweets. So, the size of the final matrix was enormous so this matrix must be reduced numbers of flouts from 64 to 16 bits by getting the mean of this matrix. The features have been extracted from the final matrix after reducing.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.4.2. Classification process

In this step, the author can be identified and classified among the other different authors and identifies if the tweet belongs to him or not. Machine learning algorithms have been applied to classify this approach including:

- Logistic Regression; which used to specify observations to a separate set of classes, converts its output using the logistic sigmoid function to return a probability value. For example, classification problems used in Email spam or not spam.
- Stochastic Gradient Descent (SGD); which used to find the values of the parameters of a function that minimizes the cost function as much as possible.
- Support Vector Machines (SVM); are a set of supervised learning models used for classification and regression.
- Support Vector Clustering (SVC); is a similar method that is also based on kernel functions but is suitable for unsupervised learning. It is the main method of data science.
- Linear Support Vector (Linear SVC); similar to SVC with parameter kernel=' linear', but implemented in terms of liblinear rather than libsvm. So, this class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.
- k-Nearest Neighbors (kNN); the neighbors are taken from a set of objects for which the class (kNN classification) or the object property value (kNN regression) is known. It's based on feature similarity approach.
- Gaussian Naive Bayes; Naive Bayes can be extended to real-valued attributes and the Gaussian (or Normal distribution) is the easiest to work with it because we need only to estimate the mean and the standard deviation from your training data.
- Decision Tree; used a tree as a model of decisions and their possible result. It is a flowchart in each input node represents a "test", each branch represents the result of the test, and each leaf node represents a class label and the paths from the root to leaf represent classification rules.
- Extreme Gradient Boosted (XGB); is a decision tree, used a more regularized model formalization to control over-fitting, which gives it better performance.
- Random Forest; creates decision trees on randomly selected data samples, gets a prediction from each tree, and selects the best solution utilizing voting. It is considered as a highly accurate and robust method. including "gini"; this is how much the model fit or accuracy decreases when you drop a variable. It is the total decrease in node impurity, "entropy"; is the measures of impurity, disorder, or uncertainty in a bunch of examples. Depending on the number of classes in your dataset.

4.5. Performance Measures Stage

To evaluate the performance of our proposed approach, we should use the basic performance measures from the confusion matrix. From the form of the confusion matrix, we will measure accuracy, precision, recall or sensitivity, F1 Score, specificity, g-mean, and ROC/AUC Curve. Table 3 consists of four outputs "True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN)" produced by a binary classifier [31].

Table 3 Confusion Matrix of Total Population

		Prediction	
		Positive	**Negative**
Observed	**True**	TP	TN
	False	FP	FN

- Accuracy: Calculated as the total number of true predictions (TP + TN) divided by the total population in (TP + FP + TN + FN).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

- Precision: It is also called a positive predictive value (PPV). Calculated as the number of true positive predictions (TP) divided by the total number of positive predictions (TP + FP).

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

- Recall (Sensitivity): It also called a True Positive Rate (TPR) or Recall (REC). Calculated as the number of true positive predictions (TP) divided by the total number of positive observed (TP+FN).

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

- F1 Score: is the harmonic mean or sub-contrary mean of Precision and Recall.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$\text{F1 Score} = 2\,x\,\frac{(\text{Recall}\,x\,\text{Precision})}{(\text{Recall} + \text{Precision})} \quad (7)$$

- Specificity: It is also called True Negative Rate (TNR). calculated as the number of true negative predictions (TN) divided by the total number of negative observed (TN+FP).

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (8)$$

- G-mean: is the geometric mean of recall and precision. It is a performance metric that calculates an sqrt of true positive rate * positive predictive value.

$$\text{G-mean} = \sqrt{\text{Recall}\,x\,\text{Precision}} \quad (9)$$

- ROC or AUC Curve: The Receiver Operating Characteristics Curve (ROC) or Area Under the Curve (AUC) can be used for visualizing and checking the classification model's performance of the multi-class classification problem. It is capable to distinguish between classes. This curve plots by two parameters: True Positive Rate (TPR); which known as sensitivity or probability of detection $\frac{TP}{TP+FN}$, versus False Positive Rate (FPR), $\frac{FP}{TN+FP}$, which known as the probability of false alarm [1-specificity].

4.6. Results and Discussion Stage

This stage displays the result of the TF-IDF vectorizer, word embedding, proposed AV with TF-IDF & word2vec approach, and the discussion.

4.6.1. TF-IDF vectorizer

To evaluate our approach, we performed some experiments including applied TF-IDF to our dataset under some conditions. From Table 4 we can show that the number of vocabularies has been used as number features; We concluded that when the features increased, the performance measurements improved.

Table 4 Results for TF-IDF Under Some Features Conditions.

Classifiers	# of features	Precision	Recall	F1-score
Logistic Regression	100 features	39%	35%	34%
	1000 features	62%	61%	61%
	10000 features	73%	72%	72%
Stochastic Gradient Descent (SGD)	100 features	29%	30%	27%
	1000 features	57%	59%	57%
	10000 features	73%	73%	73%
Multinomial Naïve Baise	100 features	37%	35%	34%
	1000 features	59%	60%	59%
	10000 features	**74%**	73%	73%
Linear Support Vector	100 features	39%	34%	33%
	1000 features	61%	60%	60%
	10000 features	**74%**	**74%**	**74%**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.6.2. Word Embedding

We also use a pre-trained word2vec model[30], trained on the Twitter dataset, which contains embeddings for unique words. After applying it on a lot of words; the distance between two words like "dog" and "fish" is indeed larger than the "dog" and "cat" distance, another important thing is the values range of the "dog" vector. Meaning that it can distinguish between words and the distance calculated is increased or decreased by gender, royal, and location. We can see, "cat" is very similar to "dog" so the distance is closed, while "fish" is not very similar to either of them, as shown in Figure 2.

```
[3.7057e-01 2.1281e-02 4.9538e-02 2.9440e-01 6.7622e-02 2.1693e+00
 2.9106e-01 2.3319e-01 1.8311e-01 5.0226e-01 1.0689e+00 1.4698e-01
 2.6748e-01 3.6462e-01 8.9088e-02 1.2243e-01 5.5095e-01 3.6410e-01
 1.5361e-01 5.5738e-01 3.8580e-01 3.8000e-01 1.4425e-01 4.9583e-02
 1.2755e-01 1.4339e-01 3.2537e-01 2.7226e-01 4.3632e-01 7.9405e-01
 2.6529e-01 1.0135e-01 4.3117e-01 1.6687e-01 1.0729e-01 8.9418e-02
 2.8635e-01 4.0117e-01 4.5217e-01 1.3521e-01 2.0224e-01 2.7184e-01
 9.1703e-02 1.8949e-01 9.2639e-02 4.3316e-01 9.8044e-01 6.7423e-01
 8.4003e-01 1.7385e-01 4.1848e-01 1.6098e-01 3.8422e-01 1.7486e-01
 5.3528e-01 2.0143e-01 3.7877e-02 4.7105e-01 1.6840e-01 3.0334e-01
 4.2730e-01 3.3803e-01 1.1343e-01 6.1958e-02 6.1808e-02 8.2018e-02
 5.1442e-02 2.8725e-01 5.8025e-01 1.0132e-01 1.4463e-01 1.1569e-02
 1.1429e-02 4.7645e-01 3.0255e-01 6.0317e-01 4.0548e-01 6.2874e-01
 6.3080e-01 2.6533e-01 2.3391e-01 1.8573e-02 3.7100e-01 2.7092e-01
 4.2409e-02 2.4656e-01 3.8445e-02 3.6120e-01 1.9113e-02 6.2741e-02
 5.8238e-01 7.8105e-01 1.0477e-02 1.0928e+00 1.0140e-01 2.5797e-01
 3.3356e-01 3.3194e-01 1.6639e-01 6.3579e-01 2.4006e-01 9.0182e-01
 1.2425e-01 4.7986e-02 2.4446e-01 4.7232e-02 3.5694e-01 4.4241e-01
 6.6037e-01 2.2759e-01 3.1055e-01 5.3650e-01 1.6563e-01 3.3707e-01
 1.0920e-01 3.7219e-01 1.4251e-01 4.1638e-01 2.1446e-01 1.8410e-01
 3.4722e-01 2.3667e-01 7.4249e-02 2.8618e-01 2.8410e-01 4.5968e-02
 3.5343e-02 2.2467e-01 5.1556e-01 9.9559e-02 2.0136e-01 2.8334e-01
 3.7766e-02 3.8784e-01 1.9552e-01 8.5009e-01 1.0345e-01 9.7010e-02
 1.1339e-01 3.9502e-01 5.9043e-02 2.1978e-01 1.8845e-01 3.3164e-01
 6.1477e-02 4.4510e-01 4.7329e-01 2.6312e-01 1.4652e-01 2.2908e-02
 1.7545e-03 5.4789e-01 3.1194e-01 2.4693e-01 2.9929e-01]
dog - cat distance : 0.19831448793411255
dog - fish distance: 0.5914566218852997
```

Figure 3 The Distance for a Pre-trained Word2vec Model.

4.6.3. TF-IDF with Word Embedding

Table 5 shows the combination of TF-IDF and word embedding in Cross-Topics datasets talking about different topics which refer to as (CT), and other Twitter datasets, which downloaded from UCI with a Single-Topic talking about healthy news, which refers as (ST)[26]. Table 5 summarizes the performance of ML algorithms that applied to our proposed approach and displays the best one. The best results achieved were marked with bold font.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 5 Result of TF-IDF with Embedding Approach in our Dataset (CS) and Health News UCI Dataset (ST).

Classifiers	Accuracy		Precision		Recall		F1-score		Specificity		G-mean	
	CT	ST	CT	ST	CT	ST	CT	ST	CT	ST	CT	ST
Random Forest Classifier"100_gini"	86%	38%	**90%**	44%	86%	38%	**87%**	35%	**99%**	**95%**	88%	37%
Random Forest Classifier"100_entropy"	85%	38%	**90**%	44%	86%	39%	**87%**	36%	**99%**	**95%**	87%	37%
Random Forest Classifier"10_gini"	85%	31%	**89%**	32%	85%	31%	86%	29%	**99%**	**95%**	87%	28%
Logistic Regression Classifier	41%	38%	45%	41%	42%	38%	42%	34%	**97%**	**95%**	43%	35%
Stochastic Gradient Descent (SGD) Classifier	38%	38%	51%	49%	39%	35%	38%	31%	**97%**	**95%**	46%	39%
Linear Support Vector Classifier	47%	42%	50%	43%	48%	43%	47%	40%	**98%**	**95%**	49%	39%
k-nearest-neighbors Classifier	50%	37%	53%	39%	50%	38%	51%	36%	**98%**	**95%**	52%	34%
Gaussian Naive Bayes Classifier	32%	29%	37%	32%	33%	29%	32%	28%	**97%**	**95%**	34%	30%
Decision Tree Classifier	84%	23%	**89**%	23%	85%	23%	86%	23%	**99%**	**94%**	86%	20%
Gradient Boosted (XGB) Classifier	61%	38%	65%	42%	62%	39%	63%	36%	**98%**	**95%**	63%	37%

As shown in Table 5 best result appear in the random forest classifier which has the best F1-score 87%in the CT dataset. The following figures have shown the Roc curve for the proposed approach after applied machine learning algorithms on our dataset CT and UCI dataset health news ST. Which had been clarified from figure 4 to figure 11 in the CT dataset, and from figure 12 to figure 19 in the ST dataset.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

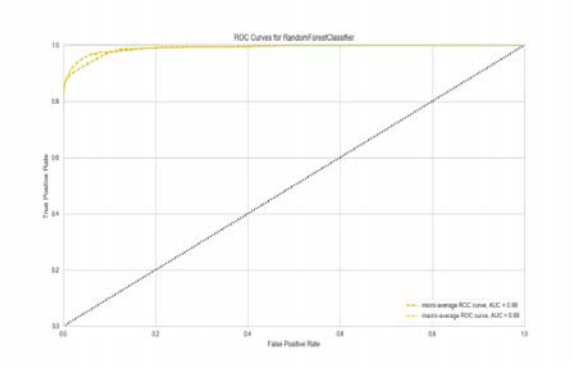

Figure 4 ROC Curve for Random Forest Classifier for Our Twitter Data.

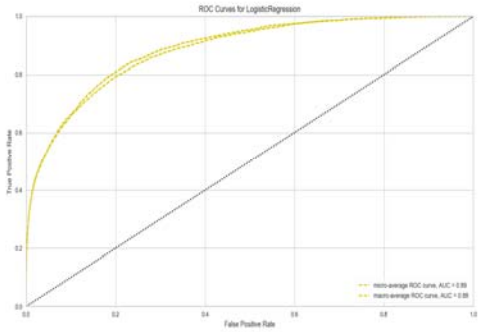

Figure 5 ROC Curve for Logistic Regression Classifier for Our Twitter Data.

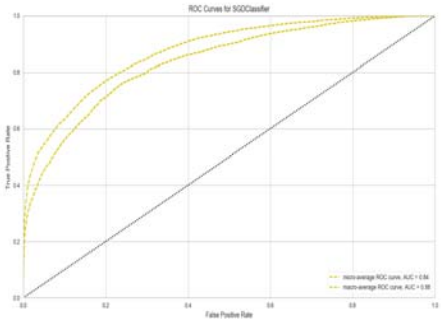

Figure 6 ROC Curve for (SGD) Classifier for Our Twitter Data.

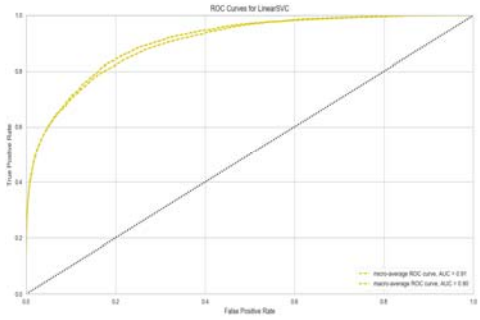

Figure 7 ROC Curve for Linear Support Vector Classifier for Our Twitter Data.

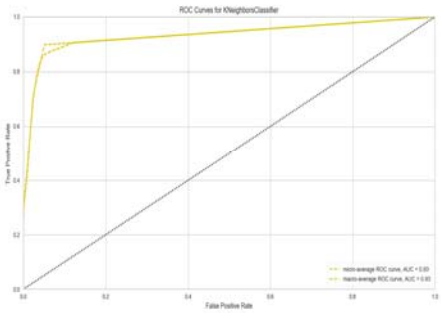

Figure 8 ROC Curve for k-NN classifier for Our Twitter Data.

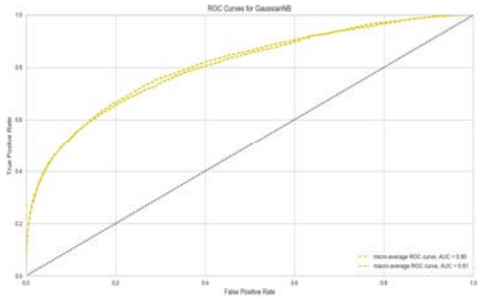

Figure 9 ROC Curve for Gaussian Naive Bayes Classifier for Our Twitter Data.

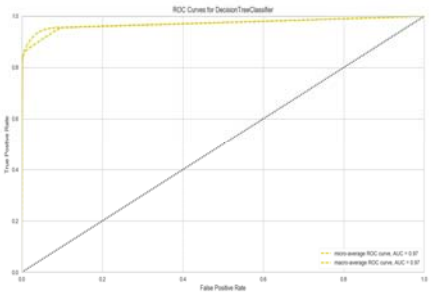

Figure 10 ROC Curve for Decision Tree Classifier for Our Twitter Data.

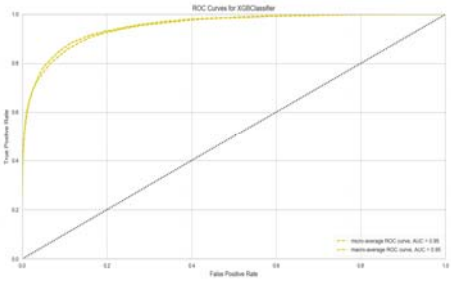

Figure 11 ROC Curve for (XGB) Classifier for Our Twitter Data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

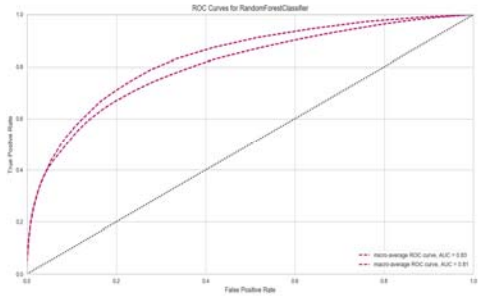

Figure 12 ROC Curve for Random Forest Classifier for UCI Dataset Health News.

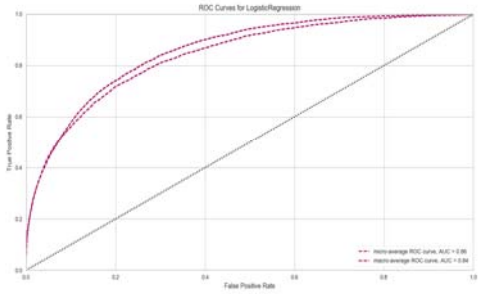

Figure 13 ROC Curve for Logistic Regression Classifier for UCI Dataset Health News.

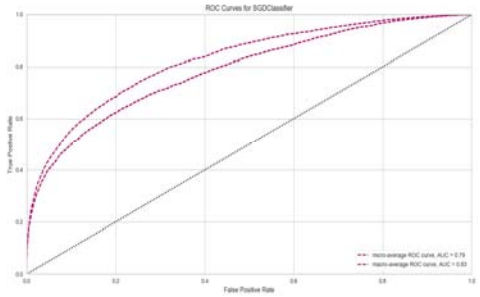

Figure 14 ROC Curve for (SGD) Classifier for UCI Dataset Health News.

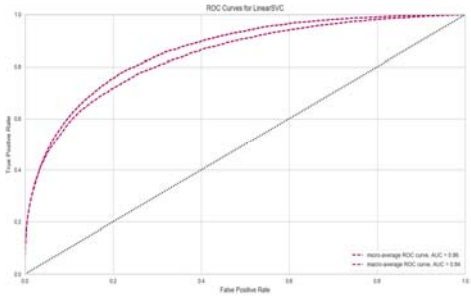

Figure 15 ROC Curve for Linear Support Vector Classifier for UCI Dataset Health News.

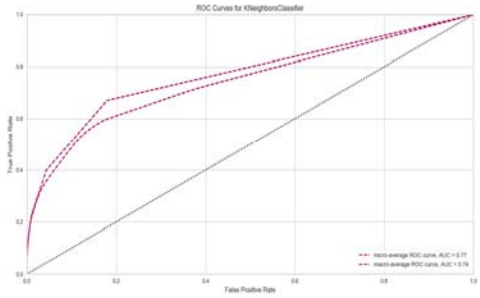

Figure 16 ROC Curve for k-NN classifier for UCI Dataset Health News.

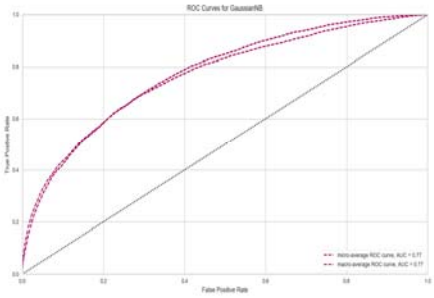

Figure 17 ROC Curve for Gaussian Naive Bayes Classifier for UCI Dataset Health News.

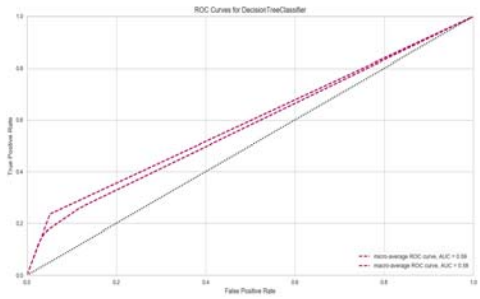

Figure 18 ROC Curve for Decision Tree Classifier for UCI Dataset Health News.

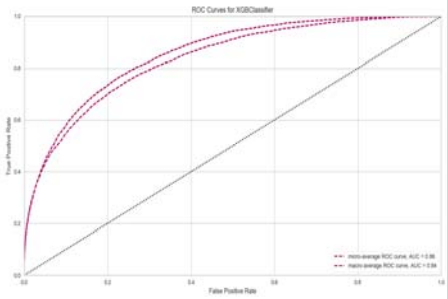

Figure 19 ROC Curve for (XGB) Classifier for UCI Dataset Health News.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.6.4. Discussion

The durability and strength of our proposed approach are in combination between vectors of TF-IDF and word embedding, also in Cross-Topics. The results were very encouraging in the dataset with different topics on single ones. This is an interesting and illuminating discussion that the proposed approach achieves promising results when compared to previous works especially in short texts. We see that the difference in results depends on the nature of different data. As we see the results of the system are excellent in the data with different subjects like our social Twitter, that have more than topic like (sports, politics, media, music, etc.) which make a proposed approach learn more new words every training and be able to distinguish the authors, and not satisfactory in the data with similar subjects like health news Twitter data that took about the health topic only. Although the subjects are similar, the accuracy and efficiency of verification are low, since there is no discrimination and distinguish the distinctive words of each author, because more than one author can speak on the same subject using the same words and thus cause confusion between the authors, and the efficiency becomes weak to distinguish between the authors.

5. CONCLUSION AND FUTURE WORK

Authorship verification is a very important task that appeared recently. It is one of the most modern areas of natural language processing which determines whether a document belongs to a particular author or not. Therefore, many types of research and attempts are being proposed in this field.

In this paper, we proposed an authorship verification approach, which depends on TF-IDF and word2vec that used the word embeddings method. Our approach can learn how to distinguish between the author's words, which depending on vectors that try to describe any close or far from the author. Some important steps are taken toward developing the performance. Our contribution is particularly useful in short texts of the different subjects of a dataset and got results in promising than a single subject. The robustness of our approach was depended on a combination of the TFIDF method and the Word Embedding method. This helps us to understand the natural language to predict the real author. The accuracy of current authorship verification technology depends mainly on the number of candidate authors, the size of texts, and the number of training texts. Finally, results showed that the proposed approach has promising results and outperform in random forest classifier.

In future work, we will improve our performance on scaling our proposed approach to work on a large number of users that require a larger resource than the ones we have, due to the vectors we generate for our data. which is very large to deal with and to do operations on our available resources. We will work to improve these limitations in future work.

REFERENCES

[1] Stein, B., Lipka, N., and Prettenhofer, P.: 'Intrinsic plagiarism analysis', Language Resources and Evaluation, 2011, 45, (1).

[2] Stein, B., Lipka, N., and zu Eissen, S.M.: 'Meta analysis within authorship verification', in Editor (Ed.) ^(Eds.): 'Book Meta analysis within authorship verification' (IEEE, 2008, edn.).

[3] Coulthard, M.: 'Author identification, idiolect, and linguistic uniqueness', Applied linguistics, 2004, 25, (4).

[4] Manevitz, L.M., and Yousef, M.: 'One-class SVMs for document classification', Journal of machine Learning research, 2001, 2, (Dec).

[5] Ehrhardt, S.: '7 Authorship attribution analysis', Handbook of Communication in the Legal Sphere, 2018, 14.

[6] Koppel, M., and Winter, Y.: 'Determining if two documents are written by the same author', Journal of the Association for Information Science and Technology, 2014, 65, (1).

[7] Potha, N., and Stamatatos, E.: 'Improving author verification based on topic modeling', Journal of the Association for Information Science and Technology, 2019

[8] Stamatatos, E.: 'A survey of modern authorship attribution methods', Journal of the American Society for information Science and Technology, 2009, 60, (3).

[9] Litvak, M.: 'Deep Dive into Authorship Verification of Email Messages with Convolutional Neural Network', in Editor (Ed.) ^(Eds.): 'Book Deep Dive into Authorship Verification of Email Messages with Convolutional Neural Network' (Springer, 2018, edn.).

[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V.: 'Scikit-learn: Machine learning in Python', Journal of machine learning research, 2011, 12,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(Oct).

[11] Li, Y., Xu, L., Tian, F., Jiang, L., Zhong, X., and Chen, E.: 'Word embedding revisited: A new representation learning and explicit matrix factorization perspective', in Editor (Ed.)^(Eds.): 'Book Word embedding revisited: A new representation learning and explicit matrix factorization perspective' (2015, edn.).

[12] Mikolov, T., Yih, W.-t., and Zweig, G.: 'Linguistic regularities in continuous space word representations', in Editor (Ed.) ^(Eds.): 'Book Linguistic regularities in continuous space word representations' (2013, edn.).

[13] Koppel, M., and Schler, J.: 'Authorship verification as a one-class classification problem', in Editor (Ed.) ^(Eds.): 'Book Authorship verification as a one-class classification problem' (ACM, 2004, edn.).

[14] Chen, X., Hao, P., Chandramouli, R., and Subbalakshmi, K.: 'Authorship similarity detection from email messages', in Editor (Ed.) ^(Eds.): 'Book Authorship similarity detection from email messages' (Springer, 2011, edn.).

[15] Brocardo, M.L., Traore, I., Saad, S., and Woungang, I.: 'Authorship verification for short messages using stylometry', in Editor (Ed.) ^(Eds.): 'Book Authorship verification for short messages using stylometry' (IEEE, 2013, edn.).

[16] AbdulRazzaq, A.A., and Mustafa, T.K.: 'Burrows-Delta method fitness for Arabic text authorship Stylometric detection', in Editor (Ed.) ^(Eds.): 'Book Burrows-Delta method fitness for Arabic text authorship Stylometric detection' (IJCSMC, 2014, edn.).

[17] Maitra, P., Ghosh, S., and Das, D.: 'Authorship Verification-An Approach based on Random Forest', arXiv preprint arXiv:1607.08885, 2016

[18] Benzebouchi, N.E., Azizi, N., Aldwairi, M., and Farah, N.: 'Multi-classifier system for authorship verification task using word embeddings', in Editor (Ed.) ^(Eds.): 'Book Multi-classifier system for authorship verification task using word embeddings' (IEEE, 2018, edn.).

[19] Mueller, J., and Thyagarajan, A.: 'Siamese recurrent architectures for learning sentence similarity', in Editor (Ed.) ^(Eds.): 'Book Siamese recurrent architectures for learning sentence similarity' (2016, edn.).

[20] Schroff, F., Kalenichenko, D., and Philbin, J.: 'Facenet: A unified embedding for face recognition and clustering', in Editor (Ed.) ^(Eds.): 'Book Facenet: A unified embedding for face recognition and clustering' (2015, edn.).

[21] Lan, M., Tan, C.-L., Low, H.-B., and Sung, S.-Y.: 'A comprehensive comparative study on term weighting schemes for text categorization with support vector machines', in Editor (Ed.)^(Eds.): 'Book A comprehensive comparative study on term weighting schemes for text categorization with support vector machines' (2005, edn.).

[22] 'TF-IDF Vectorization', pp. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[23] Potha, N., and Stamatatos, E.: 'A profile-based method for authorship verification', in Editor (Ed.) ^(Eds.): 'Book A profile-based method for authorship verification' (Springer, 2014, edn.).

[24] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., and Dean, J.: 'Distributed representations of words and phrases and their compositionality', in Editor (Ed.) ^(Eds.): 'Book Distributed representations of words and phrases and their compositionality' (2013, edn).

[25] Kenter, T., Borisov, A., and De Rijke, M.: 'Siamese cbow: Optimizing word embeddings for sentence representations', arXiv preprint arXiv:1606.04640, 2016

[26] Karami, A.: 'Health News in Twitter Data Set', 2015.https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter#

[27] GitHub: '"taspinar/twitterscraper"'. https://github.com/taspinar/twitterscraper

[28] davidmogar: ''Text normalization library for Python''. https://github.com/davidmogar/cucco

[29] McKinney, W.: 'pandas: a foundational Python library for data analysis and statistics', Python for High Performance and Scientific Computing, 2011.

[30] Yang, X., Macdonald, C., and Ounis, I.: 'Using word embeddings in twitter election classification', Information Retrieval Journal, 2018, 21, (2-3).

[31] Espíndola, R., and Ebecken, N.: 'On extending f-measure and g-mean metrics to multi-class problems', WIT Transactions on Information and Communication Technologies, 2005.