Anomaly Detection in Texts using Sentence Embeddings

Zoltán Szoplák[1,†], Abdulwahed Almarimi[2,†], Asmaa Salem[2,†] and Gabriela Andrejková[1,*,†]

[1]P. J. Šafárik University in Košice, Jesenná 5, 04001 Košice, Slovakia
[2]Bani Waleed University, Qadwar Al-Saholi 2, 00 218 322 Bani Waleed, Libya

Abstract

Many texts with unknown sources may contain so-called anomalous parts, which may have been artificially inserted from another source or author, either by mistake or on purpose. The detection of these anomalies is a vital task for detecting plagiarism, verifying authenticity, as well as cleaning texts from parts that would cause their inconsistency.

In this paper we propose a novel approach to anomaly detection in texts. The approach consists of two steps, the first step lies in detecting the potential sentences that are considered to be the borders between the anomalous and non-anomalous texts. In the second step, we pair up those sentences to find the anomalous parts. To obtain the semantic encoding of the sentences, we use a pre-built $Word2Vec$ model to individually encode the words that make up the sentence.

We propose four methods for aggregating words in sentence embedding. One of them is based on multiplying the embedded elements with its Inverse Document Frequency (*IDF*) score, another uses a special *TF-IDF* metric calculated using a Term Frequency (*TF*) and IDF. Once the sentence embeddings are obtained, a BIdirectional Long-Short Term Memory *BI-LSTM* network is used to detect sentence shifts that are on the boundaries of anomalous texts. We then use streaming semantic comparison to filter out false positives and false negatives from the boundary sentences and match them. Our approach was tested on a corpus originating from the competition External Plagiarism Detection in Arabic Text PAN 2015 (ExAraCorpusPAN2015), where we artificially replaced parts of the source text with semantically similar parts coming from another text. Our methods, based on embedding sentences obtained using Part-Of-Speech tag (*POS-Tag*) weights, achieved the highest *F1-score* value 0.95.

Keywords

anomaly detection, text dataset, word embedding, sentence embedding, autoencoder, neural networks

1. Introduction

Anomaly detection in texts is the task of identifying parts of texts which differ from other parts of the text from the point of syntax or from the point of view of semantics. The difficulty of the problem is in the fact that it is not known in advance where to look for anomalies in the text, how extensive the anomalous parts are and to what extent different parts of the text should be considered anomalies. This problem has applications in several areas, for example in detecting plagiarism, in attempts to falsify texts or in attempts to insert misleading information into texts. When solving this problem, it is necessary to follow the formal (syntax) and content (semantics) essence of the text. It is advisable to monitor both essences at the same time and not divide the problem into two tasks.

Anomaly detection is also known as outlier detection or novelty detection. Various approaches are used to solve the problem of finding anomalies in non-textual data, including: anomaly detection using deviations in high dimensional data [1], unsupervised clustering methods [2, 3], rule-based systems [4], and deep learning [5, 6, 7]. Generative Adversarial Networks (GANs) and the adversarial training process have been recently employed to solve this problem and they yield remarkable results [8, 9]. Although many anomaly detection methods are applicable to text documents, special approaches can be applied to the information contained in the texts to provide better results. Our goal was to create a sentence embedding based on word embeddings and determine the boundaries of anomalous parts of the text. Verify the implemented algorithm on Arabic texts.

In the paper in Section 2, we present some of the methods that influenced our approach to the solutions In Section 3, the proposal of the solution is elaborated, emphasizing the new elements. Section 4 is devoted to simulated datasets preparation, Section 5 contains the experimental setup, results and their evaluation. In the final Section 6, an overall evaluation of the method and a proposal for possible improvements of the presented solution are given.

2. Anomaly Detection in Texts

A new GAN-based text anomaly detection method is presented in [10], In the method, an adversarial regularized autoencoder (ARAE) is used to reconstruct normal sen-

ITAT'23: Conference on Information technologies – Applications and Theory, September 22–26, 2023, Tatranské Matliare, Slovakia
*Corresponding author.
†These authors contributed equally.
zoltan.szoplak@student.upjs.sk (Z. Szoplák); abdulwahedalmarimi@bwu.edu.ly (A. Almarimi); asmaa.salem@bwu.edu.ly (A. Salem); gabriela.andrejkova@upjs.sk (G. Andrejková)
0000-0002-2956-0496 (G. Andrejková)

tences and detects anomalies via a combined anomaly score based on the building blocks of ARAE. An overview of methods for Group Anomaly Detection can be found in [11], but to contextual anomalies are given only general attention here.

Identifying the style of the author of the given document using stylometric functions can contribute to the detection of anomalies in the texts. The described method in [12] contributes to the detection of plagiarism in texts based on the formal aspect of the text. Plagiarism in Arabic texts was solved in [13, 14]. In [15], it is presented an anomaly detection method – Context Vector Data Description (CVDD), which builds upon word embedding models to learn multiple sentence representations that capture multiple semantic contexts via the self-attention mechanism.

The introduction of a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers [16] was a significant step in language representations. BERT inspired us to look for improved solutions, however, in the article we present a solution that provides quite good results. In [17], there were used multiple machine learning methods such as sentence transformers, auto encoders, logistic regression and distance calculation methods to predict anomalies in English texts and his system gave F1-score value 0.86 for used Cross-lingual Natural Language Inference dataset plus injected Stanford Sentiment Treebank.

The paper [18], 2008, introduces the main challenges in Arabic text processing and describes the proposed unsupervised learning model for detecting anomalous Arabic textual information, but the problem is on classification if text contains some anomaly. Authors used various Arabic Web sites for text collections. Three main types of text were used to represent anomalous segments (200 - 500 words), religious description text, social text and novels. All 3 evaluation criteria (cosine similarity, city block distance and Chebychev distance measure) have achieved a significant increase in the detection rates (> 75 %) in terms of the detected anomalous segments in the Top 10 (indicates how many times each anomalous segment appeared in the top 10 of the ranked list).

In the paper [19], we solved problem: "To cover stylistic dissimilarities among text segments of the same long English or Arabic text." Using clustering and Convolutional Neural Networks (CNN) we evaluated 40 English and 40 Arabic text (using texts from pan-plagiarism-corpus-2011.part1.rar, http://ksucorpus.ksu.edu.sa) and we have got accuracy values in interval $\langle 60; 74\rangle\%$ for English texts and $\langle 53; 96\rangle\%$ for Arabic text.

Our first approach to finding positions of anomalies was using HTM networks for English and Arabic texts [20]. We achieved better results for English texts, so we now turned our attention to Arabic texts. Contextual semantic embeddings were analyzed from point of Arabic multitask classification in [21] but it doesn't solve anomalies. Our attention is focused on solving the problem of finding anomalies in Arabic texts using semantics and some stylistic features of the text.

3. Proposed Solution

In order to detect semantic anomalies from text we need (1) to find a way to encode of semantic contents of sentences, (2) to determine the position of the anomalous part of the text and (3) to evaluate the quality of the given solution.

3.1. Semantic content encoding

Since natural language text is difficult to interpret by machines, our first step is to create vector embeddings of the words and sentences that make up a given text.

To obtain the semantic content of words that make up a sentence, we can simply make use of Aravec Word2Vec [22]. Aravec is only capable of encoding singular words, therefore we need a method to create an embedding that describes the semantic content of an entire sentence. There are multiple ways to obtain sentence embeddings.

Let $S = w_1, ..., w_n$ be a sentence consisting of n words and $V = v_1, ..., v_n$ be their embeddings, where w_i represents the $i - th$ word and v_i its corresponding embedding. Suggested methods for sentence embedding are: (1) averaging of words embeddings, (2) inverse document frequency embeddings, (3) threshold of similarity embeddings, and (4) extraction of top k keyword.

3.1.1. Averaging of words embeddings

The simplest way to create a sentence embedding is to average all of its word embeddings.

$$E(S) = \frac{\sum_{i=1}^{n} v_i}{n} \tag{1}$$

While such a method may be effective, not every word contributes to the overall meaning of the sentence to the same degree.

3.1.2. Inverse document frequency embeddings

Another method lies in calculating the Inverse Document Frequency (*IDF*) value of each word from the corpus and using it to weight each word of the embedding. *IDF* was proposed by Jones [23] in 1972, and has since been extremely widely used. The *IDF* value penalizes words that are common in the entire corpus by calculating in how many documents the word is present in. It is calculated as

$$IDF(w, D) = log\frac{M}{|d \in D : w \in d|} \tag{2}$$

where D is the collection of all documents from the corpus, $|D| = M$, d is document, and $|d \in D : w \in d|$ is the number of documents that w appears in. In our case we consider every sentence a different document and calculate the *IDF* values across all sentences of all documents from our text corpus. With our *IDF* values, we can calculate *IDF* weighting based sentence embeddings

$$E(S) = \frac{\sum_{i=1}^{n} IDF(w_i, D) * v_i}{n} \tag{3}$$

3.1.3. Semantic similarity *TF-IDF* embeddings

We can also use a modified *TF-IDF* metrics (the *TF-IDF* follows two statistics, term frequency (TF) and inverse document frequency. *TF-IDF* is frequently used in a language processing [24]. The TF is relative frequency of word w within document d

$$TF(w, d) = \frac{f_{w,d}}{\sum_{w' \in d} f_{w',d}} \tag{4}$$

where $f_{w,d}$ is the number of the word w in the document d.

TF-IDF multiplies the IDF values with the term frequency metric, TF-metric

$$\textit{TF-IDF}(w, d) = TF(w, d) * IDF(w, D) \tag{5}$$

TF-metrics calculates the number of times a given term appears in the given text (not necessary full document). The TF-metric will be sampled over a window of N sentences, however it won´t be calculated over the exact matches, rather, it will be calculated over words that belong to the same context via embedding similarity.

To find the threshold of embedding similarity we can use a dictionary of synonyms as well as an thesaurus. We calculate the average cosine distance between the embeddings of every word and their corresponding synonyms from the dictionary of synonyms. Then, we calculate the average cosine distance between the embeddings of every word and their corresponding definitions from the thesaurus. Averaging these two values, we obtain the threshold embedding distance. Using this distance, we can calculate the TF values. This will ensure that the *TF-IDF* will be the words that are most relevant to distinguishing it from its neighbors. The final embedding metric is calculated as

$$E(S) = \frac{\sum_{i=1}^{n} \textit{TF-IDF}(w_i, S) * v_i}{n} \tag{6}$$

where w_i is word in sentence S.

3.1.4. *POS-Tag* relevance embeddings

We have also developed key phrase extraction method that lie in part-of-speech tagging, which assigns to each word its predicted part-of-speech tag (*POS-Tag*). Some parts of speech contribute more to the meanings of a sentence than others. Nouns, adjectives and verbs contribute more to the overall meaning than pronouns and prepositions.

We have calculated the relative weight of *POS-Tag*s by first calculating the cosine distance between the embedding of the word with a given *POS-Tag* and the overall sentence embedding gained by calculated by a simple pooling. We have averaged and normalized these values. We can then calculate sentence embeddings by multiplying the word embedding with its *POS-Tag* weight, as

$$E(S) = \frac{\sum_{i=1}^{n} \textit{POS-weight}(\textit{POS-Tag}(w_i)) * v_i}{n} \tag{7}$$

3.2. Positions of anomalous parts

After calculating the sentence embeddings, we will use some methods to detect anomalous positions. Since anomalies are generated from the same semantic cluster, simply using one-class clustering methods is inadvisable.

During the search for anomalous sentences, it is possible to use only relative relationships between sentences, or their embeddings. Let's assume that the neural network trained on the initial sentences will predict the following sentences and gradually the given text will be learned. If the predicted sentence is very different from the original sentence (we will use some metrics for comparison), then the original sentence does not fit into the text and can be considered anomalous. The problem here is in the setting of the similarity/dissimilarity threshold, which we found experimentally. Sentences that are not similar are marked as anomalous. In further processing, they are used to determine anomalous sections. However, it is possible that the anomalous section will consist of only one sentence. After that, it is still necessary to create sections using the marked anomalous sentences. Our proposed methods:

3.2.1. Anomaly thresholds in document

We can make use of a recurrent neural network, such as a Bidirectional $BI-LSTM$ network [25]. The network uses a context window of k previous sentences in order to predict the embedding of the next sentence. This prediction can then be compared to the original embedding using the cosine distance metric between its embeddings, which will give us a measurement of how well the network is able to predict the given sentence. Alternatively, the prediction success can be calculated as the prediction error of this sentence compared to the mean prediction error of the previous l sentences. Such a metric takes the predictive strength of the network into account too, as when the data is noisy a low prediction score might be

found even in a non-anomalous part of the text, however the prediction score is not that out of the ordinary.

How well the network is able to predict a given sentence is a good indicator if an anomaly threshold is found. An anomaly threshold can either be a non-anomalous sentence followed by an anomalous sentence, or in a reverse situation an anomalous sentence followed by a non-anomalous sentence. The outputs of the *BI-LSTM* network will serve as our potential anomaly thresholds.

3.2.2. Anomaly sections

Now that we have potential anomaly threshold sentences, we still need to identify which sections lying between two points are anomalous and which are not. If we merely tagged them in an alternating fashion, it would lead to a large number of errors, as a single faulty division point could mean that we misclassify our entire dataset. We need some way of determining the anomalous nature of individual sections.

We can assume that most anomalous sections are relatively short compared to non-anomalous sections. We can pair up indices of division points that are located close to one another, specifically 20 sentences from each other. While we can create possible pairings of anomalous parts that are located close to one another, there are individual division points that cannot be paired. They might be a false positive or they correspond to beginning or ending that has not been found yet. For each isolated index, we construct multiple artificial sections that are created varying distances before or after it.

Autoencoders, described in detail in [26] are unsupervised neural networks that aim to create a representation of data that selects only the most relevant parameters, which can be used to reconstruct the original data. Autoencoders consist of two main parts: the encoder, which converts the input into an encoding (usually of lesser dimension than the input), and a decoder that tries to reconstruct the input from the encoding. Using simple feed-forward neural network, the encoding h be calculated as:

$$h = \omega(Wx + b) \tag{8}$$

where x is the input, ω is an activation function, W is a weight matrix and b is the bias. This encoding can then be used to obtain x', the reconstruction of the input. The reconstruction is calculated as:

$$x' = \omega'(W'h + b') \tag{9}$$

where ω', W', b' might be different from ω, W and b.

We use the sentence embedding methods described in section 3.1 to encode every sentence of the text. We trained an deep autoencoder model on the sentences of the text. Autoencoders generalize the data they encode, and as such they can be used to detect how common certain sentence embeddings are. The idea behind this method is the following: the non-anomalous parts that make up the majority of the text come from the same source and have the same semantic makeup. Therefore an autoencoder trained on such data will be able to reconstruct them with less error than the anomalous parts that come from different sources and possess a different semantic makeup. By calculating the cosine distance of the reconstruction, we then calculate the average distance between two potential anomalous sections.

After we set the threshold for anomaly detection, we can label some sentences as anomalous and others as non-anomalous. If we find a non-anomalous sentence between two potential anomalous sections, we can use the created embeddings to compare the two anomalous sentence embeddings and if the match passes a given threshold, flag the given non-anomalous sentence as anomalous. We also look at the borders of anomalous sections.

Let $L = l_1, ..., l_n$ be the labelling for the source text $T = t_1, ..., t_n$ where l_i has two possible values, o – original and a – anomalous. After the labelling L has been created, we localize the starting point of each anomalous section (o_{i-1}, a_i), and we compare the embedding similarity (via cosine distance) of the pair o_{i-2}, o_{i-1} and the pair o_{i-1}, a_i. If the similarity between o_{i-1}, a_i was greater, we change our labelling of l_{i-1} from o_{i-1} to a_{i-1}, since otherwise the labelling remains unchanged. We also examine the relative similarities of pairs o_{i-1}, a_i and a_i, a_{i+1} and if we find that the similarity was greater in case of o_{i-1}, a_i, we change our labelling for l_i from a_{i-1} to o_{i-1}. We perform this operation until we are certain of the true anomaly border. We perform an analogous steps at the end of each anomalous section (o_{j-1}, a_j).

3.3. Evaluation criteria

Given that we know the positions of the anomalous parts in the texts of a prepared dataset, it is possible to use the classical evaluation of the achieved results, namely *accuracy, precision, recall* and *F1-Score.*

4. Dataset preparation

An evaluation of our anomaly detection method was done on dataset created from a 1000 documents from various sources written in the Arabic language. The documents range from file sizes of 5 kB containing 20 sentences to those being as long as 12728 sentences and having a file size of 3736 kB. The average sentence number of used texts is 62 sentences with an average file size of 96.3 kB.

Dataset 1: The insertion of anomalies will be an artificial process, where we will substitute small parts of texts (several sentences) from a different text of the dataset in order to create artificial anomalies. For each text, we first

decide whether the given text should be anomalous and if it is we generate a random number of anomalies, ranging from 1 to 10. Therefore, the number of anomalies in each text would range from 0 (non-anomalous, some texts are not modified) to 10 (most anomalies).

When generating an anomaly in a text, we create a sentence offset, which will mark where the anomaly should start. We then randomly choose another text from the dataset, that is different from the original one, from where we will substitute sentences. We also decide on a sentence offset in the other text, starting from which, we will substitute a randomly generated number of sentences, ranging from 2 to 20 into the original text. This substitution is only one sided, so that the sentences in the original from the given offset and a given length will be substituted by the sentences from the other text, but the other text will remain unchanged. We repeat this process for every anomaly in a given text, meaning that a given text can have substituted parts from multiple sources. After we have performed all substitutions, we save our newly anomalous text, as well as create another document for each text in which we will save the number of anomalies, and for each anomaly their starting offset and the length of that anomaly calculated in number of sentences.

Dataset 2: We have also created a modified dataset that replaces parts of texts from other semantically related texts. First, we use *TF-IDF* to extract the top 10 keywords of each texts and then use the K-means algorithm to cluster texts into semantic clusters. We use an identical process to create an anomaly text with one difference. When we create a new anomalous text, we're swapping another text from the same cluster. This ensures that two texts and by extension the swapped sentences are semantically similar, making our anomaly detection task more difficult but more relevant for real world usage.

5. Experiments

5.1. Experimental Setup

We have created 500 anomalous texts in the Arabic language to use in our prediction algorithm using methods described in the Dataset preparation section. We have implemented 4 methods of semantic sentence embeddings using Aravec Skip-Gram embeddings trained on data from Wikipedia, with a vector size of 300.

To find the given threshold of embedding similarity we have used an Arabic dictionary of synonyms as well as an Arabic thesaurus [27].

The most simple method involved simply average pooling all the non-stop word embeddings of a sentence to create our sentence embeddings. This method is described in subsection 3.1.1. and will henceforth be referred to as *Average pooling*. The second method we have implemented multiplies the word embeddings with their *POS-Tag* relevance score, before averaging them. This method is described in subsection 3.1.4. and will henceforth be referred to as *POS-Tag pooling*. The third method we have implemented multiplies the word embeddings with the IDF score of the given word before averaging them. This method is described in subsection 3.1.2. and will henceforth be referred to as *IDF pooling*. The final method we have implemented multiplies the word embeddings with a specialized *TF-IDF* score where the term frequency is calculated based on semantic similarity before averaging them. This method is described in subsection 3.1.3. and will henceforth be referred to as *TF-IDF pooling*.

We have used these sentence embeddings in conjunction with a recurrent neural network with the task of predicting the embedding of the next sentence.

The networks consists of two *BI-LSTM* layers with a $tanh$ activation function connected to a dense feed-forward layer with a $softmax$ activation function. The input of the network consists of 5 previous sentence embeddings, $k = 5$, and the task of the network is to predict the embedding of the next sentence. We chose the given value of k based on our previous experience with Arabic texts (the sentences in the texts are quite long), but we will analyze this context length in future experiments. By comparing the prediction with the ground truth we obtained anomaly scores used to mark where a given anomaly section potentially ends or begins.

The software was created in Python using available libraries for working with texts and neural networks, such as *gensim, NLTK, keras* and *tensorflow*.

5.2. Results

Sentence embeddings is the most important part of automatic semantic and syntactic analysis, therefore we evaluated several methods, four of which we describe in the article and compare their application on two simulated datasets. The simulation of the datasets consisted in inserting part of the texts from other texts, while the insertion positions are random but known for evaluation purposes. A non-trivial step is also merging anomalous sentences into continuous parts of anomalous texts (anomalous sections).

The achieved results of these four methods on two datasets can be found in Tables 1 and 2. The obtained results show that the most suitable method is the *TF-IDF* pooling for both datasets. The *TF-IDF* evaluates two statistics, term frequency and inverse document frequency. This means that it records more information about the sentences in the text.

The worst results were obtained for Average pooling, which uses the simplest approach, averaging word embeddings in a sentence. However, it is surprising that even the more complicated Dataset 2 reaches a value of

Table 1
Results of pooling methods applied on Dataset 1

Dataset 1	Accuracy	Precision	Recall	*F1 Score*
Average pooling	96.13	0.72	0.75	0.74
POS-Tag pooling	96.94	0.77	0.81	0.79
IDF pooling	97.74	0.84	0.85	0.84
TF-IDF pooling	99.23	0.96	0.93	0.95

Table 2
Results of pooling methods applied on Dataset 2

Dataset 2	Accuracy	Precision	Recall	*F1 Score*
Average pooling	93.06	0.54	0.68	0.60
POS-Tag pooling	94.68	0.61	0.74	0.67
IDF pooling	96.48	0.74	0.80	0.77
TF-IDF pooling	98.14	0.87	0.87	0.87

0.54 in the precision parameter. This means that the averaging of word embeddings contributes significantly to important information in sentence embeddings too.

Another interesting thing of note is the trade-off relationship between the precision and recall values. The Average pooling and *POS-Tag* pooling methods have a lot lower precision than recall. Meaning the algorithm tagged more sections than necessary. But the methods using IDF pooling and *TF-IDF* pooling suddenly start to change the trade-off relationship, where the precision values are approaching or even surpassing the recall values. We believe the reason for it might be the inherent ability of the IDF metric that takes into account the semantic content of other sentences. Therefore the semantic embeddings it produces are more unique.

Table 3
Sensitivity of k in regards to F1 score results on Dataset 2

Dataset 2	k=3	k=5	k=7	k=9
Average pooling	0.61	0.60	0.57	0.49
POS-Tag pooling	0.69	0.67	0.64	0.58
IDF pooling	0.75	0.77	0.77	0.76
TF-IDF pooling	0.86	0.87	0.86	0.84

We have also opted to explore the sensitivity of $F1$-score for the parameter k (the number of previous sentences) for our various pooling algorithms on Dataset 2 using the $F1$ score metric and summarized the results in Table 3. We have determined $k = 5$ as the optimal choice for our best performing pooling methods *IDF pooling* and *TF-IDF pooling*. These two methods aren't all that sensitive to change in the window size and we believe that a context window of 5 sentences is the optimal trade-off between providing just enough past information while also being capable of dealing with smaller anomalies. We observe a different trend in the embeddings created by the *Average pooling* and *POS-Tag pooling* methods, as their efficiency seems to decrease with a larger context window as they do not capture the relative uniqueness of the sentences due to lacking and *IDF* metric.

Jafari (2022) [17] showed that the outlier detector which uses a transformer based model adds more context values into anomaly detection models. The developed model was tested on Cross-lingual Natural Language Inference (XNLI) corpus which is the extension of the Multi-Genre NLI. The Stanford Sentiment Treebank (SST) is a corpus with fully labeled parse trees was used as auxiliary dataset which samples from it to be injected to XLNI dataset. A sample of 1000 English text taken from SST set injected to XNLI set to create a dataset in which XNLI sample text are normal and the SST samples are outliers. The results in the Table 4 show precision 92 %.

Table 4
Outlier Detection Results for XNLI+SST Dataset

Dataset **XNLI+SST**	Valid Sample	Precision	Recall	*F1 Score*
Jafari [17]	3490	0.92	0.828	0.86

6. Conclusion

In the article, we described four sentence embedding methods, which can be used to represent the semantic content of sentences. These alongside a recurrent neural network were used to determine the offsets of anomalous sentences. We then used the reconstruction distances of sentence embeddings from an autoencoder model trained on the entire text to determine the anomalous sections. These methods achieved good results on simulated datasets. The presented methods can provide warnings about positions of sections that could be anomalous in unknown texts.

Our next goal is to use transformers in sentence embeddings and to improve the algorithm for creating anomalous sections. Our further research will be oriented to the use of range-based performance metrics instead of point-based metrics in sentence embeddings, because a sentence is a sequence of words in a text and in point-based metrics these words are taken as a set and could occur in different permutations. Word order is important in the sentence.

Acknowledgments

The research is supported by the Slovak Scientific Grant Agency VEGA, Grant No. 1/0177/21 "Descriptional and Computational Complexity of Automata and Algorithms".

References

[1] A. Ghoting, S. Parthasarathy, M. Otey, Fast mining of distance-based outliers in high-dimensional datasets, Data Mining and Knowledge Discovery 16 (2008) 349–364. doi:10.1007/s10618-008-0093-2.

[2] A. Christy, G. Meeragandhi, S. Vaithyasubramanian, Cluster based outlier detection algorithm for healthcare data, Procedia Computer Science 50 (2015). doi:10.1016/j.procs.2015.04.058.

[3] A. Jayasimhan, J. Gadge, Anomaly detection using a clustering technique, Inter. Journal of Applied Information Systems 2 (2012) 5–9. doi:10.5120/ijais12-450391.

[4] W.-K. Wong, A. W. Moore, G. F. Cooper, M. M. Wagner, Rule-based anomaly pattern detection for detecting disease outbreaks, American Association for Artificial Intelligence (2002).

[5] J. Mu, X. Zhang, Y. Li, J. Guo, Deep neural network for text anomaly detection in siot, Computer Communications 178 (2021) 286–296. doi:https://doi.org/10.1016/j.comcom.2021.08.016.

[6] G. Pang, C. Shen, L. Cao, A. V. D. Hengel, Deep learning for anomaly detection: A review 1 (2020). doi:https://doi.org/10.48550/arXiv.2007.02500.

[7] O. Gorokhov, M. Petrovskiy, I. Mashechkin, Convolutional neural networks for unsupervised anomaly detection in text data LNCS 10585 (2017) 500–507. doi:10.1007/978-3-319-68935-7_54.

[8] F. Mattia, P. Galeone, M. Simoni, E. Ghelfi, A survey on gans for anomaly detection (2019).

[9] T. Y. Yap, Text anomaly detection with arae-anogan, Illinois Wesleyan University (2020).

[10] E. M. Billah Nagoudi, D. Schwab, Semantic similarity of arabic sentences with word embeddings v1 (2017) 18 – 24. URL: https://hal.science/hal-01683485.

[11] A. Feroze, A. Daud, T. Amjad, M. K. Hayat, Group anomaly detection: Past notions, present insights, and future prospects, SN COMPUT. SCI 2,119 (2021).

[12] A. Saini, M. R. Sri, M. Thakur, Intrinsic plagiarism detection system using stylometric features and dbscan, 2021 Inter. Conference on Computing, Communication, and Intelligent Systems (2021) 13–18.

[13] I. Bensalem, P. Rosso, S. Chikhi, A new corpus for the evaluation of arabic intrinsic plagiarism detection, CLEF 2013, LNCS 8138 (2013) 53–58.

[14] I. Bensalem, Plagiarism detection: A focus on the intrinsic approach and the evaluation in the arabic language (2020). doi:10.13140/RG.2.2.25727.84641.

[15] L. Ruff, Y. Zemlyanskiy, R. Vandermeulen, T. Schnake, M. Kloft, Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text, Proc. of the 57th Annual Meeting of the Association for Computational Linguistics (2019) 4061–4071.

[16] J. Devlin, C. Ming-Wei, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, Proc. of NAACL- HLT (2019) 4171–4186.

[17] A. Jafari, A deep learning anomaly detection method in textual data, https://arxiv.org abs/2211.13900 (2022) 8.

[18] N. Abouzakhar, B. Allison, L. Guthrie, Unsupervised learning-based anomalous arabic text detection, in: N. Calzolari (Ed.), Proc. of the Sixth International Conference on Language Resources and Evaluation, (ELRA), Marrakech, Morocco, 2008.

[19] A. Salem, A. Almarimi, G. Andrejková, Text dissimilarities predictions using convolutional neural networks and clustering*, in: J. Paralič (Ed.), Proc. of the IEEE World Symposium on Digital Intelligence for Systems and Machines, IEEE, Košice, Slovakia, 2018.

[20] Z. Szoplák, G. Andrejková, Anomaly detection in text documents using htm networks., Proc. ITAT 2021 (2021) 20–28.

[21] A. zahra El-Alami, S. O. E. Alaoui, N. E. Nahnahi, Contextual semantic embeddings based on fine-tuned arabert model for arabic text multi-class categorization, J. of King Saud University – Computer and Information Sciences 34 (2021) 8432–8428.

[22] W. Antoun, F. Baly, H. Hajj, Arabert: Transformer-based model for arabic language understanding, Proc. of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, Marseille, France, European Language Resource Association (2020) 9–15.

[23] K. S. Jones, A statistical interpretation of term specificity and its application in retrieval, J. of Documentation 28 (1972) 11–21.

[24] D. Jurafsky, J. H. Martin, Speech and language processing (3rd ed., https://web.stanford.edu/ jurafsky/slp3/14.pdf (2023) 1–29.

[25] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE TSP 45(11) (1997) 2673–2681.

[26] D. Bank, N. Koenigstein, R. Giryes, Autoencoders, CoRR abs/2003.05991 (2020). URL: https://arxiv.org/abs/2003.05991. arXiv:2003.05991.

[27] S. Alali, A. A. Sheikh, S. Al-Ahmad, The net dictionary in the arabic language, https://www. alarabimag.com/books/27725 (2019) 1–772.