mathematics

MDPI

Article

TB-BCG: Topic-Based BART Counterfeit Generator for Fake News Detection

Andrea Stevens Karnyoto [1,2,*], Chengjie Sun [2], Bingquan Liu [2] and Xiaolong Wang [2]

[1] State Key Laboratory of Communication Content Cognition, People's Daily Online, Beijing 100733, China
[2] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China; cjsun@insun.hit.edu.cn (C.S.); liubq@hit.edu.cn (B.L.); wangxl@insun.hit.edu.cn (X.W.)
* Correspondence: andre@ukitoraja.ac.id

Abstract: Fake news has been spreading intentionally and misleading society to believe unconfirmed information; this phenomenon makes it challenging to identify fake news based on shared content. Fake news circulation is not only a current issue, but it has been disseminated for centuries. Dealing with fake news is a challenging task because it spreads massively. Therefore, automatic fake news detection is urgently needed. We introduced TB-BCG, Topic-Based BART Counterfeit Generator, to increase detection accuracy using deep learning. This approach plays an essential role in selecting impacted data rows and adding more training data. Our research implemented Latent Dirichlet Allocation (Topic-based), Bidirectional and Auto-Regressive Transformers (BART), and Cosine Document Similarity as the main tools involved in Constraint @ AAAI2021-COVID19 Fake News Detection dataset shared task. This paper sets forth this simple yet powerful idea by selecting a dataset based on topic and sorting based on distinctive data, generating counterfeit training data using BART, and comparing counterfeit-generated text toward source text using cosine similarity. If the comparison value between counterfeit-generated text and source text is more than 95%, then add that counterfeit-generated text into the dataset. In order to prove the resistance of precision and the robustness in various numbers of data training, we used 30%, 50%, 80%, and 100% from the total dataset and trained it using simple Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). Compared to baseline, our method improved the testing performance for both LSTM and CNN, and yields are only slightly different.

Keywords: fake news detection; Latent Dirichlet Allocation (LDA); Bidirectional and Auto-Regressive Transformers (BART); cosine document similarity; AAAI2021-COVID19 Fake News Detection dataset

Citation: Karnyoto, A.S.; Sun, C.; Liu, B.; Wang, X. TB-BCG: Topic-Based BART Counterfeit Generator for Fake News Detection. *Mathematics* **2022**, *10*, 585. https://doi.org/10.3390/math10040585

Academic Editors: Andrea Prati, Carlos A. Iglesias, Luis Javier García Villalba and Vincent A. Cicirello

Received: 22 December 2021
Accepted: 10 February 2022
Published: 14 February 2022

1. Introduction

Social media brings collective intelligence by sharing knowledge between users. News on social media that contains reviews, opinions, and other information has played an essential role in people's decisions [1]. It also has a significant effect on our social life. At the same time, the development of social media technologies raises adverse effects, such as the spread of fake news. Fake news and misleading articles are straightforward to circulate because they have particular traction [2]. The appealing thing about fake news is that sometimes authors write the information using clickbait titles, bombastic narration, and viral topics discussion. One characteristic of news articles published in social media is often their lesser quality as compared to conventional news since there is no regulatory authority on social media [3]. There are several types of fake news, such as fake rumors, satires, fake reviews, advertisements, misleading content, fake speech about politics, and many more. Compared to mainstream media, social media circulates fake news faster, contributing to increased partisan conflict and political polarization [4,5]. Fake news has been spreading intentionally and misleading society to believe false information; this phenomenon has made identifying fake news based on shared content very difficult.

Information on social media is actively disseminated from person to person, causing a particular content detection problem. Fake news circulation is not only a current issue; it has been disseminated for centuries. Dealing with fake news is a complicated task due to its massive spread. Therefore, automatic fake news detection is needed.

Fake news detection research has become necessary due to the limitation of human ability to handle massive news spread on the internet. Additionally, researchers can develop systems that help people select information for their consumption. Many studies have been done in this field, including (1) an automated system for distinguishing fake news [6], (2) analysis by time-series in different domains for sentiment predictive discussion method [7], (3) capturing steady differences in the language of fake and real news by using deep neural networks [8], and (4) building a shared convolutional neural network using two paths, which are shared low-level features and joint optimization [9], and many more.

In addition to being used to detect fake news [10,11], natural language processing has been widely utilized for various more sophisticated tasks, i.e., Human Activity Recognition (HAR) [12], sentiment analysis [13–15], and spoken notifications for intelligent environments [16]. This paper aims for involvement in the Constraint @ AAAI2021-COVID19 Fake News Detection dataset shared task using the pre-trained model and deep learning. Our proposed method implemented Latent Dirichlet Allocation (Topic-based), BART, and Cosine Document Similarity as the primary tools for the COVID-19 dataset. This method fills a lack of training datasets by selecting a dataset based on topic and creating a counterfeit generated training dataset to increase the model's accuracy. A topic model is one of the research fields in computer sciences, especially in text mining and information retrieval tasks. It uses the machine learning method to induce a generative probabilistic model of text corpora [17]. Latent Dirichlet allocation (LDA) is the most reputable topic model, introduced in 2003 by Blei et al. [18]. According to Arjovsky et al. and Hou et al. [19,20], small datasets create problems, such as (1) the discriminator is always overfitting for the training dataset, (2) the feedback becomes insignificant when it sends into the generator, and (3) training data start to deviate. However, an augmentation generator that generates adversarial data can be applied to the training dataset; it will help to increase the number of dataset rows [21]. Furthermore, it can improve performance (accuracy, precision, and F1-score) although modifying training data sometimes will decrease the natural composition of information [22,23]. We created a counterfeit training dataset using BART with OpenAI GPT2 as pre-trained. BART is an autoencoder that can reduce the noise for pretraining sequence-to-sequence models [24,25]. In fact, BART is made by training many corrupted documents and optimizing the cross-entropy's regeneration loss function between the decoder's output and the original document.

The main ideas of this paper include:

1. We initiated involvement in Constraint @ AAAI2021-COVID19 Fake News Detection by integrating transfer learning, selecting a training dataset using the topic model, and generating a counterfeit training dataset.
2. The main idea of this paper is to select the most impacted data using topic-based methods and then sort it based on data distinctiveness. Moreover, we generated counterfeit training data using BART and compared it toward source text using cosine similarity. Counterfeit-generated text is created from 70% source text and 30% automatically generated text. If the comparison cosine similarity result between counterfeit generated text and source text is more than 95%, add counterfeit generated text into the dataset.

In order to prove the resistance of performances and the robustness in various numbers of data training, we experimented with various dataset sizes (30%, 50%, 80%, and 100%) for random rows, random + generated text, topic-based, and TB-BCG. Then, we tested those datasets by using simple Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) models to obtain comprehensive results.

This article is presented as follows: Section 2 is the related work to review the LDA as a topic model, Bidirectional and Auto-Regressive Transformers (BART) in terms of generating

text, and Cosine Document Similarity. Section 3 states the framework and methodology in theoretical and practical, which contains six processes: data processing, LDA, adding training documents using the text generator, text generator, document similarity, and models. Section 4 contains hardware and software specifications in experiment setup and the experiment steps in description of task. Section 5 is the result. Section 6 is the discussion and future work, and Section 7 is the conclusion section.

2. Related Work

2.1. AAAI2021-COVID19 Fake News Detection

In this COVID-19 pandemic, direct (face-to-face) communication has decreased. Millions of us express our ideas only through social media, used for publishing any information freely on the Internet. Furthermore, The Constraint@AAAI2021-COVID19 Fake News Detection was created to determine whether the information regarding COVID-19 is fake or real. ***Combating Online Hostile Posts in Regional Languages during Emergency Situation*** held a first workshop, where they created COVID-19 Fake News Detection in English dataset. They collected data from many social media platforms, such as Facebook, Instagram, Twitter, etc. The dataset contains 10,700-row real and fake news articles from social media posts related to COVID-19. Many techniques are applied towards this dataset: Shushkevich et al. [26] constructed the ensemble consisting of Logistic Regression, Support Vector Machine, Naive Bayes, Bidirectional Long Short-Term Memory, and a combination of Naive Bayes and Logistic Regression. Glazkova et al. [27] made a COVID-Twitter-BERT (CT-BERT) model based on the transformer-based ensemble. They worked on an ensemble approach of different pre-trained language models, such as Ernie, Roberta, BERT, etc. Li et al. [28] created various training strategies, including k-fold cross-validation, learning rate schedule, and warm-up. Gautam and Masud [29] proposed combining LDA to topical distributions with contextualized representations from XLNet. However, we applied different approaches and techniques in this research using the LDA topic model and BART. LDA will automatically select training datasets from various topics, which can make a significant impact even when using a small training dataset. At the same time, BART helps generate additional samples to add more text into the training dataset. This approach provided a patchwork of augmentation to cover the lack of training data.

2.2. Topic-Based

Scientists and many companies met the problem of arranging millions of articles in many cases, and this is not filled with unmanageable searches. With statistical tools, we can automatically arrange electronic archives to facilitate users for efficient exploration and browsing. Topic models play an essential role in computer science, especially text mining. The topic model generates a list of words (text) and then agglomerates them into a topic group using statistical methods. A text can be various, and it can be a book chapter, an email, a journal article, a blog post, and any other miscellaneous text. We can choose the number of topics depending on the additional latent variable analysis and a persistent problem in topic modeling. For other cases, external data sources determine the number of topics and make them part of the problem formulation. Moreover, applying such collections requires a structured process: find articles similar to other articles and explore the collection by covering potential topics. Topic models are "[probabilistic] latent variable models of documents that exploit the correlations among the words and latent semantic themes" [18]. The purpose of LDA is to develop a model that appears from various topics, where the topic is determined as a distribution of constant vocabulary. One of the excellent textual analysis methods grounded in computational linguistics research is LDA. This method calculates the statistical correlations among terms/words in a massive set of texts to distinguish and quantify the underlying topics in these documents [30]. With assumption, all topics are related to collection, and all documents contain topics in different proportions. This perception is also a natural assumption because documents in corpus tend to be heterogenous; combining subsets from multiple topics will increase

the connection between topics and corpus. LDA was widely used to solve many cases; Daniel Maier et al. [31] developed a brief, hands-on user guide for applying LDA topic modeling n communication research. Wang, Y et al. [32] proposed a deep learning model for automobile insurance fraud detection that uses Latent Dirichlet Allocation (LDA)-based text analytics. Zamani et al. [33] proposed a dynamic, content-specific LDA topic modeling technique that can help to identify different domains of COVID-specific discourse that can be used to track societal shifts in concerns or views. Gurcan and Cagiltay [34] proposed a semi-automatic methodology for the semantic analysis of online job advertisements using LDA to discover the hidden semantic structures from a given textual corpus. Wiedemann et al. [35] employed a sequentially combined BiLSTM-CNN neural network to improve the classification performance with background knowledge with topic clusters obtained by LDA. Jelodar et al. [30] discussed topic modeling and then reviewed it in various fields, such as medical and linguistic science, political science, and software engineering. However, our approach employed LDA to select documents in the dataset based on data distinctiveness.

2.3. Bidirectional and Auto-Regressive Transformers (BART)

We used BART to generate additional text for training data, and it makes sense since we face obstacles to getting similar data to increase testing accuracy. BART has a primary purpose: an autoencoder that can reduce the noise of corrupted maps documents to derive the original documents. BART was implemented as a sequence-to-sequence model and applied with a bidirectional encoder over corrupted text and a left-to-right autoregressive decode as the main tools. BART utilizes transformer architecture with the standard sequence-to-sequence algorithm [36]. Text generation can be developed using machine translation. Liu et al. [37] showed significant performance from many machine translation (MT) tasks that utilized multilingual denoising pre-training. Other research generates text using a simple method. Roller et al. [38] employed a standard Seq2Seq transformer architecture to generate responses rather than retrieve them from a fixed set based on the ParlAI version and Byte-Level BPE tokenization.

2.4. Cosine Document Similarity

Weighing pairwise document similarity is essential for numerous text applications, including nearest neighbor search, document filtering, and document clustering. Moreover, we can define two documents as near-duplicate using a similarity measure as a quantitative measure [39]. Most of the similarity measures estimate the similarity between two documents based on the information content and their term weights that are shared in common [40]. The cosine measure is the most widespread measure based on the Vector Space Model (VSM) for evaluating document similarity purposes. Thousands of attributes can represent a document, each attribute recording the term-frequency of a particular word (such as a keyword) or phrase in the document. Thus, a term-frequency vector describes each document as an object. The Vector Space Model generates a space in documents and interprets those as vectors. An appropriate document description model is required to collect text documents, such as in the common Vector Space Model (VSM) [41]. By cosine document similarity, we find out which is the most distinctive data row towards other rows by calculating the level of similarity each row.

3. Proposed Method

According to Figure 1, the process starts from preprocessing; this process aims to clean text, convert emojis, expand words, and spelling correction. Select data based on topics; this process aims to classify and distribute documents according to topic, then sort documents based on topic scores. Results are four datasets, with sizes 30%, 50%, 80%, and 100% of the overall dataset. Next, generate a counterfeit document using BART based on original text to make various text. After that, calculate similarity using the cosine similarity function if the similarity is more than 95%, then add counterfeit as new data. Last, the final step is to train a simple LSTM or CNN to obtain the final result.

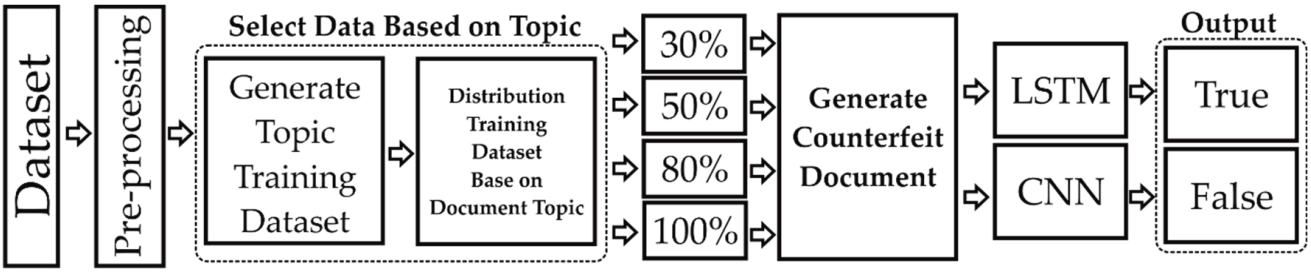


Figure 1. Overall Process of TB-BCG.

Data distribution of a dataset. Table 1 shows a balanced data distribution for training, validation, and testing. The amount of validation and testing is 33% of the training dataset for each. In addition, the number of real and fake news has the same number of data, and the distribution of unique words is not far apart.

Table 1. Data distribution of a dataset.

Data	Real	Fake	Total	Unique Word
Training	3360	3060	6420	30,046
Validation	1120	1120	2140	13,697
Testing	1120	1120	2140	14,121

3.1. Data Processing

First, we executed our tweet preprocessing and text preprocessing for transformer-based models by removing useless punctuation marks for text classification. We kept symbols @ and # because those have specific semantics in tweets. Second, we transformed the text into lowercase and replaced URLs, mentions, and emojis into unique tokens. Third, we used the Python emoji library to change the particular emoji into a meaningful textual description: redheart:, thumbsup:, etc. Furthermore, we converted hashtags into words ("#COVID"→"COVID"). Removing stopwords was not applied for this dataset to increase the ability to generate new text. We did expand contractions into full words, for example, "you're" to "you are." We also performed spell correction in processing textual data by using TextBlob. This library gives a compatible API that can change uncomplete words to common words. It can increase natural language processing (NLP) accuracy, such as noun phrase extraction, sentiment analysis, text classification, part-of-speech tagging, and more. Last, we turned all letters to lowercase.

3.2. Latent Dirichlet Allocation (LDA)

To select documents in the dataset based on the LDA method, we used the Gensim module. Gensim provides an LDA module that is highly scalable, strong, good performance-optimized, and well tested by many users. We utilized LDA multicore Online Learning, which uses whole CPU cores in a computer to parallelize and speed up model training.

Suppose we have four documents containing specific words/terms, then split each document into words and count it by number, as shown in Table 2.

Table 2 shows that each document has terms/words contained in the document. Because LDA uses a bag-of-words mechanism, we should count every term in a document, for example, Doc 1 has no terms "Corona", one term "India", two terms "Disease", and so on.

Table 2. Document Vector or Term-Frequency Vector.

	Term								
Document	**Corona**	**India**	**Disease**	**Lung**	**Disaster**	**Virus**	**Case**	**Test**	**People**
Doc 1	0	1	2	1	1	0	0	0	1
Doc 2	2	1	1	0	0	0	1	0	0
Doc 3	1	2	1	2	0	0	1	0	0
Doc 4	1	1	1	2	1	2	2	0	2

During this work, we use the same symbol and terminology used by David Blei [18], Hamed Jelodar et al. [30], and Ponweiser [17]. LDA states that each document can be represented as a probabilistic distribution over latent topics and that topic distribution in all documents' topic distribution shares a common Dirichlet prior. Each latent topic in the LDA model is also represented as a probabilistic distribution over words, and the word distributions of topics share a common Dirichlet prior.

According to David Blei [18], a word is the basic unit of discrete data, defined to be an item from a vocabulary indexed by $(\{1, \ldots, V\})$. A corpus is a collection of M documents denoted by D. Document d having N_d words $(d \in \{1, \ldots, M\})$, LDA models D according to the following generative process [30]:

(1) Choose a multinomial distribution φ_t for topic t $(t \in \{1, \ldots, T\})$ from a Dirichlet distribution with parameter β.
(2) Choose a multinomial distribution θ_d for document d $(d \in \{1, \ldots, M\})$ from a Dirichlet distribution with parameter α.
(3) For a word w_n $(n \in \{1, \ldots, N_d\})$ in document d,
    (a) Select a topic Z_n from θ_d.
    (b) Select a word w_n from φ_{zn}.

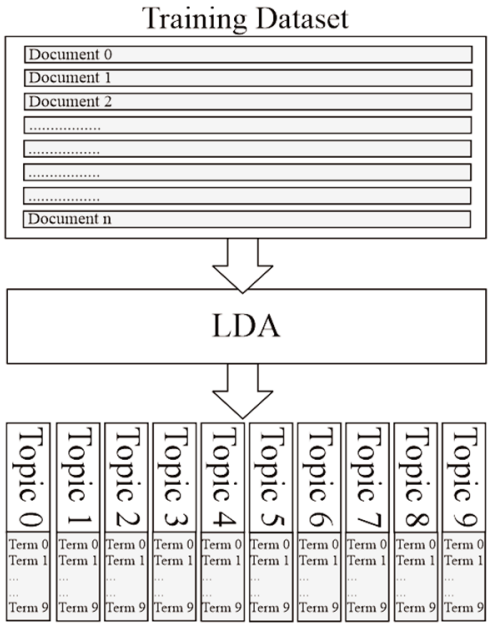
The word in documents is only an observed variable during the generative process with $(\varphi$ and $\theta)$ as latent variables, where φ is word distribution for topic t, and θ document topic distribution for document d. $(\alpha$ *and* $\beta)$ are hyperparameters, where α is Dirichlet prior parameter of per-document-topic distribution, and β is Dirichlet prior parameter of per topic–word distribution. To obtain these latent variables and hyperparameters, the probability of the observed data D is calculated and maximized as follows [30]:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha)\left(\prod_{n=1}^{N_d} \sum_{Z_{dn}} p(Z_{dn}|\theta_d)p(w_{dn}|Z_{dn}, \beta)\right)d\theta_d \ . \quad (1)$$

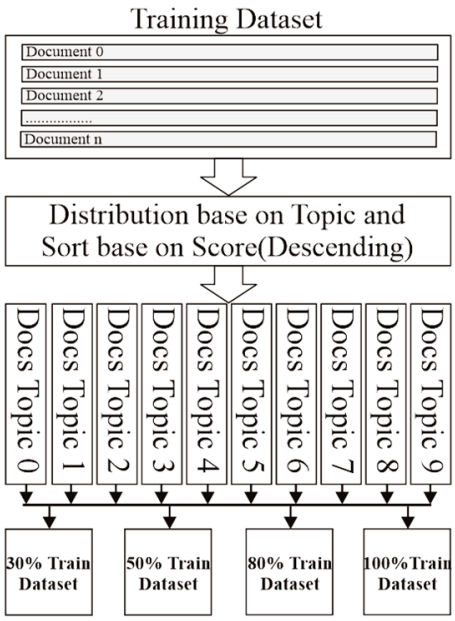
Defined, β is drawn from the Dirichlet distribution, given α are the distribution of words over topics and parameters of topic Dirichlet prior. Defined, M is some documents, N is the vocabulary size, and T is the number of topics. The Dirichlet-multinomial pair for the corpus-level topic distributions is considered as (α, θ). The Dirichlet-multinomial pair for topic-word distribution is given as (β, φ). The variable θ_d are document-level variables sampled per document. Z_{dn}, w_{dn} variables are word-level variables and are sampled for each word in each text-document. As a result, each document is then seen as a probability distribution over set of topics.

3.3. Selecting and Deviding Dataset

We divided the process into two steps in order to select documents for training dataset using the LDA mechanism that we already explained in a previous subsection: (1) LDA model generates training topics to produce ten topics. Each topic consists of 10 terms. Then, (2) distribute documents in training dataset based on same LDA model in step 1; this process aims to classify and distribute documents based on the topic, then sort documents based on topic scores. The steps are shown in the Figure 2a,b.



(**a**) Generate Topic Training Dataset



(**b**) Distribution Training Dataset Base on Document Topic

Figure 2. (**a**) The generate process diagram aims to produce topics using documents from the training dataset. (**b**) Using the same LDA model in Figure 1, the system will distribute documents in the dataset based on topic and sort documents based on scores in descending order.

3.4. Sort Most Distinctive Row in Dataset

In order to find out which is the most distinctive data row towards other rows, we need to calculate the level of similarity of a row toward other rows. We do this to each topic group. The equation can be seen below:

$$Sim_{(x)} = \sum_{x=0}^{n} \sum_{y=0}^{n} Similarity\left(text_{(x)}, text_{(y)}\right) \quad . \tag{2}$$

$Sim_{(x)}$ is the similarity value of each row in the dataset, n is the amount of documents, and x and y are iteration variables. $text_{(x)}$ and $text_{(y)}$ are the variables to be compared. After counting each row, based on the results of the calculations using the equation above, we sorted in ascending order—the more considerable the similarity value, the more similar the text row towards others. The explanation regarding text similarity can be found in sub-Section 3.7 of this paper.

The final result of the process is four datasets that contain 30%, 50%, 80%, and 100% documents from each topic. For illustration, the 30% training dataset indicates that dataset consists of the top 30% of documents classified as topic_0, the top 30% of documents classified in topic_1, and so on.

3.5. Add Training Documents Using the Text Generator

We also added generated documents into the training dataset. The Algorithm 1 can be seen as follows:

Algorithm 1. Document generation and labeling

```
input:
d: document
label: label of document (0 = fake, 1 = true)
output:
g: generated document
new_label: new label of document
def generate_text(d, label)
n = length(d) # count length of text d
q = d [0:int(n*0.70)] # get the 70% of text
p = generate(q, num_of_results = 50, lenght_of_result = n)
# Generating 50 counterfeit documents
for i = 0 to num_of_results: # iteration from 0 to 49
p_sim[] = similarity(d, p[i]) # get similarity by
# comparing each counterfeit
# text(p[i]) toward original
# document(d)
t = argmax(p_sim) # find the highest similarity
# for current text
if p_sim[t] > 0.95: # if the similarity more 95%
new_label = label # then classified same with
# the original document (d)
return g = p[t], new_label # the results is
# generated document
# and new label
```

This function has two inputs, namely *d* = document (consist *n* words) and *label* = label of document (0 = fake, 1 = true). Output of this function is *g* = generated document and *new_label* = label of generated document.

Where *n* is the length of the document, *q* contains 70% number of words from the *d* document. Variable *p* is an array variable containing generated documents (produces 50 (fifty) documents); each generated document has the same length as the d document. Furthermore, we obtain the similarity score between each generated document and d document using cosine similarity, then put values into the *p_sim* array variable. Variable *t* is the Argmax of the *p_sim* array variable. We decided that if the similarity score exceeds 0.95, then the *new_label* for the generated document (*g*) is equal to label, and we then save the generated text as counterfeit-generated text.

We chose 70% of the original text to generate 30% for the rest because this combination still keeps more than half the actual text, and the sentence does not lose too much meaning. Previously, we tested several combinations of 50% original text to 50% generated text, but it is too difficult to gain 95% similarity; moreover, we tested 85% original text to 15% generated text, but the meaning is too close between the original text and counterfeit. Hence, we decided to use 70% of the original text to generate 30% for the rest.

3.6. Text Generator

Algorithm 1 shows one of the processes is to generate ten documents, then select the one closest to the original. We utilized the GPT2 model as the transformer text-generator tool, and we used a low-level API, namely the pipeline. This model uses the "Language Models are Unsupervised Multitask Learners." It applied BART as a model combination. Pretraining has two stages: (1) arbitrary noising function for the many corrupted texts and (2) reconstructing the original text using a sequence-to-sequence model for the learned process. BART has an autoregressive decoder for Sequence Generation Process, and the system can directly fine-tune it. After that, the system will copy and manipulate all information to perform denoising to move closer to the pre-training objective. Here, the decoder generates outputs autoregressively, and the input encoder is the input sequence in stage (1).

To generate new sentences, it combines five methods as seen in Figure 3: for token masking, BERT implements randomized tokens, makes them sampled, and replaces them with (MASK) elements. For token deletion, delete random tokens in text input. This process is different from token masking; the model itself decides which places are missing in the inputs. For text infilling, take samples using several text spans, then replace every span with a single (MASK) token. If spans are 0-length, then correspond to the insertion of (MASK) tokens. For sentence permutation, a document is distributed into several sentences based on total stop words and then rearranged in random order. For document rotation: chose a token uniformly and randomly, then rotate the document and put that chosen token at the beginning of the document. Those five methods will generate the following words one by one until the specified sentence length is reached. The input is a sequence of words. Then, the process of token masking, token deletion, sentence permutation, document rotation, and text infilling are executed. The results of the execution of these processes are combined, and the process decides the most appropriate word to become the generator's result. Table 3 shows original document and a few samples of counterfeit generated document.

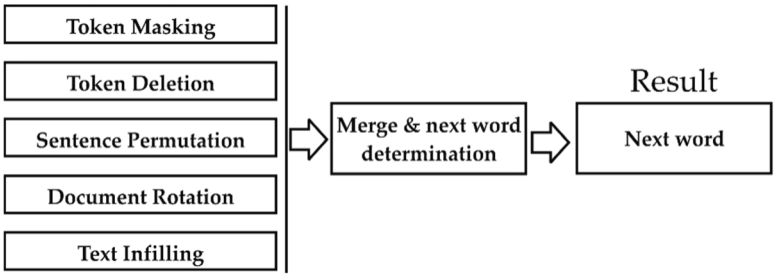


Figure 3. BART Transformations for next-word generator.

Table 3. Examples of Counterfeit Generated Document.

Original Document	There are people isolating in the Auckland quarantine facility from the community which includes people who have tested positive for COVID and their household contacts.
Counterfeit Generated Document	There are people isolating in the Auckland quarantine facility from the community which I'm speaking of, a small number of who have gone through the process to become isolated, but who I'm sure have developed the qualities required for.
	There are people isolating in the Auckland quarantine facility from the community which, as a safety precaution, we do not believe can be properly monitored", she said. A community trust group called New Zealand Community Life.
	There are people isolating in the Auckland quarantine facility from the community which will likely leave the community for the next week because of a problem with the disease", says Shue. The National Council will also be working.
	… … …
	There are people isolating in the Auckland quarantine facility from the community which has a medical staff under them. It's a big isolation center and there is no way for them to get help from the public. So, on Friday.

3.7. Document Similarity

We used cosine similarity to get the distance between two documents. Those documents that we compared are original documents toward each generated document in the array. In many NLP tasks, a cosine similarity is an approach to measuring the similarity between two non-zero vectors, aiming to get the inner product space. A document can contain thousands of attributes/words; the characteristic of the document depends on attributes, and we can classify it as the frequency based on a particular word/keyword/phrase. This method is called a term-frequency vector, for which the main task is to represent text into objects. For example, we have two documents and then generate both into term-frequency vectors. However, those vectors have many zero values in common, meaning that both vectors have no corresponding documents and do not share many words. As a result, those texts are not similar. In this process, the measurement will focus on the words that the two documents have in common and words frequency. In other words, it is necessary to measure for numeric data that overlooks zero matches.

The formula of similarity function is following:

$$Similarity(x, y) = \frac{x \cdot y}{||x||||y||} \tag{3}$$

where $||x||$ is the Euclidean norm of vector $x = (x_1, x_2, x_3, \ldots, x_n)$ or defined as $\sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2}$. In theory, it is the length of the vector. Furthermore,$||y||$ is the Euclidean norm of vector y. This similarity equation calculates the distance of vectors x and y, and the result is the cosine of the angle. If the cosine value of 0, the two vectors are at 90 degrees to each other (orthogonal), or we can say those vectors have no match.

3.8. Models

To obtain comprehensive results, we tested our approach to two simple models, which are based on Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN).

First, we focused on the RNN based duo to the LSTM network as an RNN since the RNN is a more straightforward system. The model used Facebook/Bart-large pre-trained as input; it produces matrix 300 $\times$ 1024 for their result. Then, we utilized an LSTM network, and 2 Dense (Relu) as shown in Figure 4. This simple network can efficiently use past input features via an LSTM layer and forward the information to the following Dense Layers. For the last layer, we used the Sigmoid function to obtain binary output (0 = fake, 1 = true).

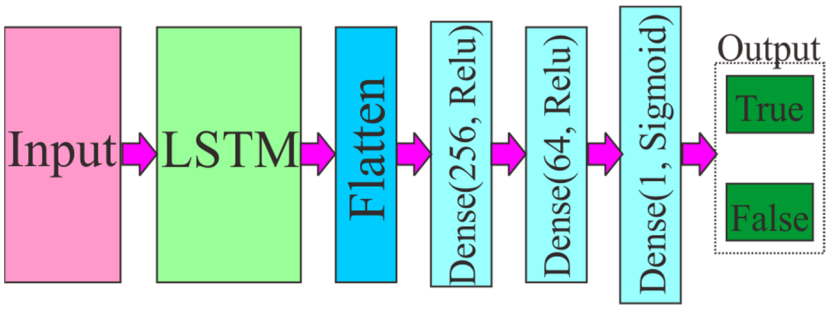


Figure 4. Simple LSTM-Based model for binary classification.

Second, we utilized Convolutional Neural Network 1-d. It is a classical approach to fix a simple problem, such as the ensembles of decision trees, training machine learning models, and time-series data on fixed-sized windows. In this research, we used the same matrix dimension and pre-processing as input. Those two models produce binary classification by using the sigmoid function. Figure 5 shows that all components in the structure are like Figure 4 except the Conv1D layer.

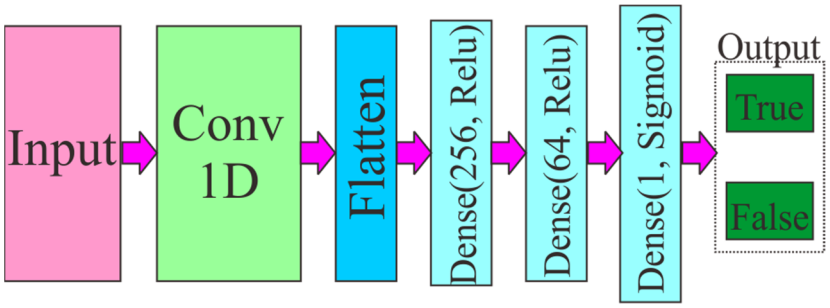


Figure 5. Simple CNN-Based model for binary classification.

4. Experiment and Task

4.1. Experiment Setup

The experiments were run on Intel Core (TM) i7 8700, 6 core 3.20 GHz Processor, 16 GB RAM, Nvidia GeForce RTX 3060 GPU 12 GB. Base program and training used Python 3.7.8 and TensorFlow Version: 2.4.0. Pre-training and tokenizer used BART transformer 3.4.0 with PyTorch 1.7.1 + cu101. Gensim 3.8.0 for LDAMulticore, TextBlob 0.15.3, tweet-preprocessor 0.6.0. Several important hyper-parameters determined this architecture: Training model learning-rate = 0.001, epoch 10, batch-size 8.

4.2. Description of Task

To gain comprehensive results and prove our proposed method was accurate in various training dataset sizes, we tested models for the different dataset sizes. First, we took random 30, 50, 80, and 100 percent of the total dataset rows as a baseline. Those baseline datasets are referred to as Train-30, Train-50, Train-80, and Train-100, respectively. Second, we took datasets based on the topic-selected method in the same size as the baseline and named those Topic-30, Topic-50, Topic-80, and Topic-100. Third, for datasets that use text generation, we defined them as TB-BCG-30, TB-BCG-50, TB-BCG-80, and TB-BCG-100. We preprocessed the generation text as explained in the proposed model method in this paper. Furthermore, we tested each model (LSTM and CNN) toward the testing dataset, and the results can be seen in the next section.

5. Result and Analysis

In this section, we show and analyze the results we obtained according to section experiment and task and discuss topics and their weight, topic distribution in datasets, statistics, percentage, and counterfeit-generated text in datasets, and the result would be to give the accuracy of each model.

First, we randomly divided the dataset into four datasets as a baseline and named those as Train-30 (30% from the dataset), Train-50, Train-80, and Train-100. It is necessary because it will produce results that can compare to our method more comprehensively.

Second, we generated counterfeit text for every train random data (Train-50, Train-80, and Train-100) by using the GPT2 model as the transformer text-generator tool. We named those as Train-gen-30, Train-gen-50, Train-gen-80, and Train-gen-100.

Third, we implemented LDA to extract topics from the original training Constraint @ AAAI2021-COVID19 Fake News Detection dataset. The ten topics and their top 10 words are presented with relative weights. Unlike Guo et al. [42] and Hao et al. [43], we did not name each topic based on the relevant relationship between their top words and their identical relative weights. As shown in Table 3 for the example, we used Topic 0, 1, 2, . . . , 9.

Table 4 shows terms and their weight in the topic, and Gensim generates it automatically. Another significant contribution LDA made in statistical distribution theory is weighted distributions, elaborating the two-level models in the traditional hierarchical Bayesian literature. LDA estimates that each text comes from multiples topics and returns the weight of each topic for each document. LDA succeeds in fixing both problems by

handling the topic mixture weights as a k-parameter hidden random variable rather than a large set of individual parameters explicitly linked to the training set. Because The LDA algorithm does not return the same results since it incorporates some randomness to the initialization process, we executed topics until we obtained well-distributed topic distribution between topics.

Table 4. Topic Terms and Their Weight (* means "for term/word").

Topic	Words
0	0.036*"deaths" + 0.030*"amp" + 0.027*"states" + 0.018*"cases" + 0.016*"numbers" + 0.014*"pradesh" + 0.011*"days" + 0.011*"spread" + 0.010*"coronavirus" + 0.009*"day"
1	0.075*"cases" + 0.030*"total" + 0.029*"health" + 0.028*"number" + 0.019*"active" + 0.019*"new" + 0.017*"world" + 0.016*"report" + 0.014*"recovered" + 0.014*"deaths"
2	0.048*"coronavirus" + 0.044*"cases" + 0.034*"reported" + 0.032*"new" + 0.020*"states" + 0.017*"risk" + 0.016*"death" + 0.015*"patients" + 0.013*"today" + 0.013*"deaths"
3	0.059*"people" + 0.050*"coronavirus" + 0.017*"cases" + 0.014*"hospital" + 0.013*"says" + 0.011*"case" + 0.010*"new" + 0.010*"contact" + 0.009*"north" + 0.009*"government"
4	0.038*"tests" + 0.034*"testing" + 0.023*"people" + 0.022*"day" + 0.016*"positive" + 0.015*"amp" + 0.015*"today" + 0.014*"state" + 0.013*"test" + 0.012*"data"
5	0.037*"new" + 0.036*"cases" + 0.030*"virus" + 0.018*"vaccine" + 0.015*"coronavirus" + 0.014*"says" + 0.013*"trump" + 0.012*"pandemic" + 0.012*"people" + 0.011*"restrictions"
6	0.060*"cases" + 0.052*"new" + 0.028*"deaths" + 0.027*"confirmed" + 0.027*"states" + 0.022*"reported" + 0.020*"lagos" + 0.020*"fct" + 0.017*"discharged" + 0.017*"tests"
7	0.023*"coronavirus" + 0.022*"data" + 0.019*"positive" + 0.019*"new" + 0.019*"test" + 0.018*"tested" + 0.017*"trump" + 0.016*"president" + 0.016*"people" + 0.015*"pandemic"
8	0.035*"tests" + 0.028*"number" + 0.020*"completed" + 0.018*"total" + 0.015*"new" + 0.015*"coronavirus" + 0.015*"yesterday" + 0.014*"learn" + 0.014*"amp" + 0.012*"vaccine"
9	0.055*"coronavirus" + 0.038*"india" + 0.018*"video" + 0.015*"testing" + 0.015*"hospital" + 0.013*"people" + 0.012*"died" + 0.011*"cases" + 0.011*"claim" + 0.011*"novel"

The most interpretable topics extracted from the fake news and real news are shown in Table 4. We divided it into ten topics, and each topic has ten words. As the result for topic 0, the words and their weight are 0.036*"deaths" + 0.030*"amp" + 0.027*"states" + 0.018*"cases" + 0.016*"numbers" + 0.014*"pradesh" + 0.011*"days" + 0.011*"spread" + 0.010*"coronavirus" + 0.009*"day". This means the top 10 keywords that contribute to this topic are: "deaths", "amp", "states", and so on, and the weight of "deaths" on topic 0 is 0.036. The number of hidden topics is a fundamental arrangement in topic modeling, and the weights reflect how important a keyword is to that topic. This LDA method represents the marginal distribution of a document as a continuous mixture combination. The observed data are the words of each text, and the hidden variables represent the latent topical structure, i.e., the topics themselves and how specific composition presents them.

Table 4 reveals that in this research dataset, coronavirus and words related to it are the most frequently appearing words. All sequence in topics have slightly diverse words, and the difference is only in order of words and the weight. It proved that both fake news and real news could have the same topics. Although we will not decide whether it is true using

the topic model, dividing it into a few topics will help us know the correlation between one text and another.

Fourth, we obtained the topics assigned to each document with the method topic distributions of the LDAModel. We created four datasets with different amounts of documents based on topics shown in Table 4, namely 30%, 50%, 80%, and 100% of the total training dataset. Table 5 shows the total number of documents and how many documents there are for each topic.

Table 5. Topics Distribution in Datasets (Bold text is highest number of topic group for each dataset).

Dataset	Topic										Total
	0	1	2	3	4	5	6	7	8	9	
Topic-30	151	171	156	216	208	211	111	209	161	**256**	1850
Topic-50	253	286	261	360	347	352	185	349	268	**427**	3088
Topic-80	404	458	417	576	555	563	296	559	429	**683**	4940
Topic-100	506	573	522	720	694	704	371	699	537	**854**	6180

As shown in the results presented in Table 5, Topic 9 has the most significant number of documents, and Topic 6 is the smallest. Because we distributed topics into datasets according to the percentage of documents number in total, it creates a balanced distribution in each dataset. The Topic-100 dataset means we retrieve the entire dataset, but it is sorted differently from the actual dataset.

Fifth, we generated counterfeit text by using the GPT2 model as the transformer text-generator tool. Our model used the "Language Models are Unsupervised Multitask Learners". We also applied pre-training BART. Table 6 shows the difference for each dataset, statistics, percentage, and counterfeit-generated text.

Table 6. Statistics, Percentage, and Counterfeit-Generated Text in Datasets.

Dataset	Statistics			Percentage			Generated Text		
	Total Row	True	False	True (%)	False (%)	Difference	True	False	Total
Train-30	1925	1003	922	0.5210	0.4790	0.0421	-	-	-
Train-50	3209	1667	1542	0.5195	0.4805	0.0390	-	-	-
Train-80	5134	2676	2458	0.5212	0.4788	0.0425	-	-	-
Train-100	6418	3360	3058	0.5235	0.4765	0.0471	-	-	-
Train-gen-30	3218	1255	1063	0.5414	0.4585	0.0828	252	141	393
Train-gen-50	3887	2103	1784	0.5410	0.4589	0.0820	436	242	678
Train-gen-80	6166	3341	2825	0.5418	0.4581	0.0836	665	308	973
Train-gen-100	7801	4081	3720	0.5231	0.4768	0.0462	721	662	1383
Topic-30	1844	921	923	0.4994	0.5005	0.0010	-	-	-
Topic-50	3082	1569	1513	0.5090	0.4909	0.0181	-	-	-
Topic-80	4935	2537	2398	0.5140	0.4859	0.0281	-	-	-
Topic-100	6418	3360	3058	0.5235	0.4765	0.0471	-	-	-
TB-BCG-30	2488	1368	1120	0.5498	0.4501	0.0996	447	197	644
TB-BCG-50	3986	2214	1772	0.5554	0.4445	0.1108	645	259	904
TB-BCG-80	6295	3495	2800	0.5552	0.4447	0.1104	958	402	1360
TB-BCG-100	7828	4082	3746	0.5214	0.4785	0.0429	892	760	1652

The results presented in Table 6 show that Train-30, Train-Gen-30, Topic-30, and TB-BCG-30 have different total rows. The difference between Train-30 and Topic-30 is caused due to the way we divided rows. For Train-30, it has more rows because we divided it directly from total data, which is 30% from the entire rows. Meanwhile, for topic-30, we obtained 30% of each topic, as seen in step four in this section. The generated texts caused Topic-Gen-30 to obtain more rows. However, this process made the dataset between true

and fake unbalanced. Take as an example TB-BCG-30; these additions of generated text enlarged the difference between true and fake, which is 9.9% and is contrary to Train-30 and Topic-30, which have a difference of 4.21% and 0.1%, respectively. Not all rows in the dataset produced the generated text as a result. If the similarity level between real text and generated text is less than 95%, the system will not add it as counterfeit-generated text. TB-BCG-30 only generated 644 counterfeit-generated texts from a total of 1844 Topic-30 rows.

Sixth, we trained all the datasets using the simple LSTM and CNN models to evaluate our model, just as we described in Section 3. All our experiments used the same hyperparameters. We utilized four different metrics, including precision, accuracy, recall, and F1-score, to compare the results of each dataset. Because F1-score represents recall and precision, we take F1-score as a performance comparison. The results can be seen in Tables 7–10 for 30%, 50%, 80%, and 100% of the dataset, respectively. Bold texts are the highest performance (F1-Score) in each table.

Table 7. Experimental result of 30% dataset.

Dataset	LSTM				CNN			
	Pre.	**Acc.**	**F1-Score**	**Recall**	**Pre.**	**Acc.**	**F1-Score**	**Recall**
Train-30	0.8701	0.8977	0.8830	0.8963	0.8317	0.7664	0.8310	0.8304
Train-gen-30	0.8981	0.969	0.8992	0.9003	0.909	0.9752	0.9100	0.911
Topic-30	0.8417	0.7764	0.8410	0.8404	0.8868	0.8597	0.8860	0.8853
TB-BCG-30	0.9100	0.9411	**0.9104**	0.9109	0.917	0.9106	**0.9167**	0.9165

Table 8. Experimental result of 50% dataset.

Dataset	LSTM				CNN			
	Pre.	**Acc.**	**F1-Score**	**Recall**	**Pre.**	**Acc.**	**F1-Score**	**Recall**
Train-50	0.9230	0.9064	0.9148	0.9068	0.8697	0.8095	0.8680	0.8664
Train-gen-50	0.9168	0.9804	0.9177	0.9187	0.9368	0.9516	0.9370	0.9372
Topic-50	0.8797	0.8195	0.8780	0.8764	0.8748	0.9384	0.8758	0.8769
TB-BCG-50	0.9359	0.9578	**0.9362**	0.9365	0.9415	0.929	**0.9411**	0.9408

Table 9. Experimental result of 80% dataset.

Dataset	LSTM				CNN			
	Pre.	**Acc.**	**F1-Score**	**Recall**	**Pre.**	**Acc.**	**F1-Score**	**Recall**
Train-80	0.9175	0.9217	0.9194	0.9214	0.9319	0.9489	0.9321	0.9324
Train-gen-80	0.9450	0.9322	**0.9446**	0.9443	0.943	0.9475	**0.9431**	0.9433
Topic-80	0.9289	0.9036	0.9283	0.9278	0.9271	0.9347	0.9271	0.9272
TB-BCG-80	0.9360	0.9364	0.9359	0.9359	0.9372	0.9225	0.9368	0.9364

Table 10. Experimental result of 100% dataset.

Dataset	LSTM				CNN			
	Pre.	**Acc.**	**F1-Score**	**Recall**	**Pre.**	**Acc.**	**F1-Score**	**Recall**
Train-100	0.9257	0.9193	0.9225	0.9194	0.9209	0.8956	0.9203	0.9198
Train-gen-100	0.9441	0.9463	0.9441	0.9441	0.9373	0.973	0.9378	0.9384
Topic-100	0.9399	0.9569	0.9401	0.9404	0.9305	0.9136	0.9300	0.9296
TB-BCG-100	0.9608	0.9732	**0.9609**	0.9611	0.9568	0.9684	**0.9569**	0.9571

We compared all results based on a group of amounts row. Tables 7, 8 and 10 show that our approach obtained the highest F1-score results for groups 30%, 50%, and 100% of the dataset. That means our proposed method worked well here. By rearranging and selecting the dataset based on their distinctiveness, we can provide a more diverse order. However, for Table 9, Train-gen-80 gained the best result. We assume this happens because data distribution is balanced between fake and true labels. The training dataset obtained from the selected topic (Topic-30, Topic-50, Topic-80, Topic-100) obtained worse performance than the Train-gen (Train-gen-30, Train-gen-50, Train-gen-80, Train-gen-100) because the topic data are fewer than the Train-gen. TB-BCG affected accuracy due to the number of rows that were successfully generated. However, TB-BCG yields the highest performance in our experiment. LSTM and CNN obtain performance results that were quite close between them.

6. Discussion and Future Work

Although our approach was successful in many experiments in this paper, many things can still be improved. For example, in LDA, we manually determined the number of topics and the number of terms in each topic. We tried to add a topic or terms, but performances are worse than the baseline. Thus, topics and terms number can be determined by the dataset and the amount of data. The experimental result of 80% dataset shows that Train-gen-80 demonstrated the best performance; this indicates that our approach cannot surpass the baseline under certain circumstances. Generating using Algorithm 1 and the BART text generator still has many shortcomings. The lack of generated text categorized in the criteria (similarity $\geq$ 95%) makes the unequal dataset distribution between true and false labels; according to Table 6, the generated text only makes up less than 50 percent of the total data.

To overcome the difficulties and drawbacks found in this work, we suggest two research directions:

First, develop a new method using LDA but with a system that automatically determines the number of topics and the number of terms for each topic. It will be helpful to make the system better in reading the dataset and performing the necessary configurations to obtain a more promising performance.

Second, create a new robust algorithm so that even the new words change up to 30% by random throughout the text but still have similarities to the original text.

7. Conclusions

This paper uses the pre-trained model and deep learning to involve the Constraint @ AAAI2021-COVID19 Fake News Detection dataset shared task. Our proposed method implemented Latent Dirichlet Allocation (Topic-based), Bidirectional and Auto-Regressive Transformers, and Cosine Document Similarity as the main tools (TB-BCG). We generated counterfeit training data using BART and compared it toward source text using cosine similarity. Counterfeit-generated text is created from 70% source text and 30% automatically generated text. These additions of generated text enlarged the difference between true and fake, which for TB-BCG-30 is 9.9%. Contrary to Train-30 and Topic-30, the difference is only 4.25% and 0.1%, respectively. In our cases, not all rows can produce the counterfeit-generated text. Therefore, as a result, the difference between true and fake is 4%. The training dataset obtained from the selected topic unsignificant increase accuracy because of the lack of diversity between true and fake. Moreover, the training dataset obtained from the selected topic decreased accuracy if compared to the Train-gen group. However, it still increases compared to the training group, and it is because we rearranged and selected the dataset based on its distinctiveness. TB-BCG affected accuracy due to the number of rows that were successfully generated. However, TB-BCG yields the highest accuracy in our experiment. LSTM and CNN obtained accuracy results that were quite close between them.

Author Contributions: Conceptualization, A.S.K. and C.S.; methodology, A.S.K.; software, A.S.K.; validation, C.S., B.L. and X.W.; formal analysis, A.S.K.; investigation, A.S.K.; resources, C.S.; data curation, B.L.; writing—original draft preparation, A.S.K.; writing—review and editing, C.S., B.L.

and X.W.; visualization, A.S.K.; supervision, C.S., B.L. and X.W.; project administration, B.L.; funding acquisition, C.S. and B.L. All authors have read and agreed to the published version of the manuscript."

Funding: Supported by State Key Laboratory of Communication Content Cognition, People's Daily Online (No. A12003).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the anonymous reviewers for their insightful comments and suggestions.

Conflicts of Interest: The authors (Andrea Stevens Karnyoto, Chengjie Sun, Bingquan Liu, Xiaolong Wang) of the paper (Title: TB-BCG: Topic-Based BART Counterfeit Generator for Fake News Detection) declare that there is no conflict of interests.

References

1. Ozbay, F.A.; Alatas, B. Fake news detection within online social media using supervised artificial intelligence algorithms. *Phys. A Stat. Mech. Its Appl.* **2020**, *540*, 123174. [CrossRef]
2. Ahmed, H.; Traore, I.; Saad, S. Detection of online fake news using n-gram analysis and machine learning techniques. In *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*; Springer: Cham, Switzerland, 2017; pp. 127–138.
3. Shu, K.; Mahudeswaran, D.; Wang, S.; Lee, D.; Liu, H. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data* **2020**, *8*, 171–188. [CrossRef] [PubMed]
4. Tandoc, E.C., Jr.; Lim, Z.W.; Ling, R. Defining "fake news" A typology of scholarly definitions. *Digit. Journal.* **2018**, *6*, 137–153.
5. Kaliyar, R.K.; Goswami, A.; Narang, P.; Sinha, S. FNDNet–a deep convolutional neural network for fake news detection. *Cogn. Syst. Res.* **2020**, *61*, 32–44. [CrossRef]
6. Buntain, C.; Golbeck, J. Automatically identifying fake news in popular twitter threads. In Proceedings of the 2017 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 3–5 November 2017; pp. 208–215.
7. Kursuncu, U.; Gaur, M.; Lokala, U.; Thirunarayan, K.; Sheth, A.; Arpinar, I.B. Predictive analysis on Twitter: Techniques and applications. In *Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining*; Springer: Cham, Switzerland, 2019; pp. 67–104.
8. O'Brien, N.; Latessa, S.; Evangelopoulos, G.; Boix, X. The Language of Fake News: Opening the Black-Box of Deep Learning Based Detectors. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, QC, Canada, 3–8 December 2018.
9. Dong, X.; Victor, U.; Chowdhury, S.; Qian, L. Deep two-path semi-supervised learning for fake news detection. *arXiv* **2019**, arXiv:1906.05659.
10. Karnyoto, A.S.; Sun, C.; Liu, B.; Wang, X. Transfer learning and GRU-CRF augmentation for COVID-19 fake news detection. *Comput. Sci. Inf. Syst.* **2021**, 53. [CrossRef]
11. Karnyoto, A.S.; Sun, C.; Liu, B.; Wang, X. Augmentation and heterogeneous graph neural network for AAAI2021-COVID-19 fake news detection. *Int. J. Mach. Learn. Cybern.* **2022**, 1–11. [CrossRef]
12. Nayak, S.; Panigrahi, C.R.; Pati, B.; Nanda, S.; Hsieh, M.Y. Comparative analysis of HAR datasets using classification algorithms. *Comput. Sci. Inf. Syst.* **2022**, *19*, 47–63. [CrossRef]
13. Chen, J.; Becken, S.; Stantic, B. Lexicon based Chinese language sentiment analysis method. *Comput. Sci. Inf. Syst.* **2019**, *16*, 639–655. [CrossRef]
14. Ljajic, A.; Marovac, U. Improving sentiment analysis for twitter data by handling negation rules in the Serbian language. *Comput. Sci. Inf. Syst.* **2019**, *16*, 289–311. [CrossRef]
15. Trisna, K.W.; Jie, H.J. Deep Learning Approach for Aspect-Based Sentiment Classification: A Comparative Review. *Appl. Artif. Intell.* **2022**, 1–37. [CrossRef]
16. Šoić, R.; Vuković, M.; Ježić, G. Spoken notifications in smart environments using Croatian language. *Comput. Sci. Inf. Syst.* **2021**, *18*, 36. [CrossRef]
17. Ponweiser, M. *Latent Dirichlet Allocation in R*; Vienna University of Economics and Business: Vienna, Austria, 2012.
18. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
19. Arjovsky, M.; Bottou, L. Towards principled methods for training generative counterfeit networks. *arXiv* **2017**, arXiv:1701.04862.
20. Hou, L.; Kong, W.; Gao, Y.; Chen, Y.; Li, X. PA-GAN: Graph Attention Network for Preference-Aware Social Recommendation. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; Volume 1848, p. 012141.
21. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]

22. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to counterfeit attacks. *arXiv* **2017**, arXiv:1706.06083.
23. Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; Jordan, M. Theoretically principled trade-off between robustness and accuracy. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 7472–7482.
24. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv* **2019**, arXiv:1910.13461.
25. Reimers, N.; Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv* **2019**, arXiv:1908.10084.
26. Shushkevich, E.; Cardiff, J. TUDublin team at Constraint@ AAAI2021-COVID19 Fake News Detection. *arXiv* **2021**, arXiv:2101.05701.
27. Glazkova, A.; Glazkov, M.; Trifonov, T. g2tmn at Constraint@ AAAI2021: Exploiting CT-BERT and ensembling learning for COVID-19 fake news detection. *arXiv* **2020**, arXiv:2012.11967.
28. Li, X.; Xia, Y.; Long, X.; Li, Z.; Li, S. Exploring text-transformers in aaai 2021 shared task: COVID-19 fake news detection in english. *arXiv* **2021**, arXiv:2101.02359.
29. Gautam, A.; Masud, S. Fake news detection system using XLNet model with topic distributions: CONSTRAINT@ AAAI2021 shared task. *arXiv* **2021**, arXiv:2101.11425.
30. Jelodar, H.; Wang, Y.; Yuan, C.; Feng, X.; Jiang, X.; Li, Y.; Zhao, L. Latent Dirichlet allocation (LDA) and topic modeling: Models, applications, a survey. *Multimed. Tools Appl.* **2019**, *78*, 15169–15211. [CrossRef]
31. Maier, D.; Waldherr, A.; Miltner, P.; Wiedemann, G.; Niekler, A.; Keinert, A.; Pfetsch, B.; Heyer, G.; Reber, U.; Häussler, T.; et al. Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. *Commun. Methods Meas.* **2018**, *12*, 93–118. [CrossRef]
32. Wang, Y.; Xu, W. Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud. *Decis. Support Syst.* **2018**, *105*, 87–95. [CrossRef]
33. Zamani, M.; Schwartz, H.A.; Eichstaedt, J.; Guntuku, S.C.; Ganesan, A.V.; Clouston, S.; Giorgi, S. Understanding weekly COVID-19 concerns through dynamic content-specific LDA topic modeling. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, Online, 16–20 November 2020; Volume 2020, pp. 193–198.
34. Gurcan, F.; Cagiltay, N.E. Big Data Software Engineering: Analysis of Knowledge Domains and Skill Sets Using LDA-Based Topic Modeling. *IEEE Access* **2019**, *7*, 82541–82552. [CrossRef]
35. Wiedemann, G.; Ruppert, E.; Jindal, R.; Biemann, C. Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. *arXiv* **2018**, arXiv:1811.02906.
36. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaizer, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
37. Liu, Y.; Gu, J.; Goyal, N.; Li, X.; Edunov, S.; Ghazvininejad, M.; Lewis, M.; Zettlemoyer, L. Multilingual Denoising Pre-training for Neural Machine Translation. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 726–742. [CrossRef]
38. Roller, S.; Dinan, E.; Goyal, N.; Ju, D.; Williamson, M.; Liu, Y.; Xu, J.; Ott, M.; Smith, E.M.; Boureau, Y.-L.; et al. Recipes for building an open-domain chatbot. *arXiv* **2020**, arXiv:2004.13637.
39. Gunawan, D.; Sembiring, C.A.; Budiman, M.A. The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents. *J. Physics: Conf. Ser.* **2018**, *978*, 012120. [CrossRef]
40. Thongtan, T.; Phienthrakul, T. Sentiment classification using document embeddings trained with cosine similarity. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, Florence, Italy, 28 July–2 August 2019; pp. 407–414.
41. Ristanti, P.Y.; Wibawa, A.P.; Pujianto, U. Cosine similarity for title and abstract of economic journal classification. In Proceedings of the 2019 5th International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 23–24 October 2019; pp. 123–127.
42. Guo, Y.; Barnes, S.; Jia, Q. Mining meaning from online ratings and reviews: Tourist satisfaction analysis using latent dirichlet allocation. *Tour. Manag.* **2017**, *59*, 467–483. [CrossRef]
43. Hao, H.; Zhang, K.; Wang, W.; Gao, G. A tale of two countries: International comparison of online doctor reviews between China and the United States. *Int. J. Med. Inform.* **2017**, *99*, 37–44. [CrossRef] [PubMed]