electronics

Article

Drug Adverse Event Detection Using Text-Based Convolutional Neural Networks (TextCNN) Technique

Ashish Rawat [1], Mudasir Ahmad Wani [2,*], Mohammed ElAffendi [2], Ali Shariq Imran [3], Zenun Kastrati [4] and Sher Muhammad Daudpota [5]

1 Koantek, Chandler, AZ 85224, USA
2 EIAS Data Science Lab, Prince Sultan University, Riyadh 11586, Saudi Arabia
3 Department of Computer Science (IDI), Norwegian University of Science and Technology (NTNU), 2815 Gjøvik, Norway
4 Department of Informatics, Linnaeus University, 35195 Växjö, Sweden
5 Department of Computer Science, Sukkur IBA University, Sukkur 65200, Pakistan
* Correspondence: mwani@psu.edu.sa; Tel.: +966-(0)-567-389-413

Abstract: With the rapid advancement in healthcare, there has been exponential growth in the healthcare records stored in large databases to help researchers, clinicians, and medical practitioner's for optimal patient care, research, and trials. Since these studies and records are lengthy and time consuming for clinicians and medical practitioners, there is a demand for new, fast, and intelligent medical information retrieval methods. The present study is a part of the project which aims to design an intelligent medical information retrieval and summarization system. The whole system comprises three main modules, namely adverse drug event classification (ADEC), medical named entity recognition (MNER), and multi-model text summarization (MMTS). In the current study, we are presenting the design of the ADEC module for classification tasks, where basic machine learning (ML) and deep learning (DL) techniques, such as logistic regression (LR), decision tree (DT), and text-based convolutional neural network (TextCNN) are employed. In order to perform the extraction of features from the text data, TF-IDF and Word2Vec models are employed. To achieve the best performance of the overall system for efficient information retrieval and summarization, an ensemble strategy is employed, where predictions of the selected base models are integrated to boost the robustness of one model. The performance results of all the models are recorded as promising. TextCNN, with an accuracy of 89%, performs better than the conventional machine learning approaches, i.e., LR and DT with accuracies of 85% and 77%, respectively. Furthermore, the proposed TextCNN outperforms the existing adverse drug event classification approaches, achieving precision, recall, and an F1 score of 87%, 91%, and 89%, respectively.

Keywords: text summarization; abstractive and extractive summarization; text-based CNN; logistic regression; decision tree; TF-IDF; Word2Vec; text classification

check for updates

Citation: Rawat, A.; Wani, M.A.; ElAffendi, M.; Imran, A.S.; Kastrati, Z.; Daudpota, S.M. Drug Adverse Event Detection Using Text-Based Convolutional Neural Networks (TextCNN) Technique. *Electronics* **2022**, *11*, 3336. https://doi.org/10.3390/electronics11203336

Academic Editors: Sebelan Danishvar, Morad Danishvar and Seyed Naser Razavi

Received: 11 September 2022
Accepted: 9 October 2022
Published: 17 October 2022

1. Introduction

Due to rapid advancements in healthcare, there has been exponential growth in healthcare records which are stored in large databases, such as PUBMED and MEDLINE, to help researchers and clinicians to seek information for optimal patient care, research, and trials. The information includes reports, records, scans study and abstracts. Because these studies are lengthy and time consuming, new information retrieval methods have arisen to make the process easier. One such solution is text summarization [1]. By allowing to obtain the gist of any given topic, automatic text summarization can help clinicians and researchers to seek information efficiently. Unlike summaries produced by humans, summaries generated automatically are less biased, efficient, decrease reading time, and are considered optimal for question-answering systems. The abstractive and extractive models can be utilized for text summarization.

Extraction of key portions in the text and derivation of a subset of the original text's preserving the verbatim is known as "extractive summarization" [2], whereas, in order to create a new, shorter text that effectively communicates the most relevant information from the original text, a state-of-the-art technique evolved that interprets and analyzes the text to replicate key material in a new way known as "abstractive summarization" [3].

Since adverse drug reactions (ADRs), which are one of the main causes of a particular illness (morbidity), and the loss of lives as a result of that particular illness (mortality) [4] are a risk to patients, it has been seen that it is very complex to detect several ADRs because they affect certain groups of people in specific situations and can take a long time to reveal. The adverse drug event classification (ADEC) module is in charge of classifying and detecting adverse drug reactions. Prior to selling the drugs, healthcare professionals engage in clinical trials to identify ADRs, but their numbers are often small. As a result, once the medications have been exhausted on the market, post-market drug safety monitoring is necessary to aid in the identification of ADRs. Spontaneous reporting systems (SRSs) (https://www.intechopen.com/chapters/63595, accessed on 17 May 2022) is the official channel supported by the Food and Drug Administration in the United States. However, many ADRs are not recorded in these systems, and these systems are frequently under reported. Recently, unstructured data, such as medical reports [5], were utilized to detect content with ADRs.

Furthermore, daily case reports are abundantly created and published in the biomedical literature on a regular basis. Not only case reports, but online social networks are also sources of useless and unwanted data that are also in an unstructured format. Sometimes, Tweets or Facebook posts containing ADRs may not be clinically important, but a huge volume of such content can reveal serious or unknown repercussions.

In a recent study [6], clinical researchers evaluated how effectively medical professionals could distinguish between allergies and intolerances, categorize the severity of ADRs, and determine the degree of contraindications. Health experts were asked to characterize the reaction as allergy or intolerance, rate the severity of the reaction, and assess the level of contraindication of the causing medicine based on the online questionnaire for ADR situations. The number and proportion of responses were calculated for each of the cases presented, and associations between the classification of reaction type, severity, and level of contraindication were examined. Some studies, such as [7,8], employed the convolutional recurrent neural network (CRNN) for ADR classification. Likewise in [7], they applied the attention layer to CNN and concatenated it to RNN. The convolutional neural network with attention (CNNA) is produced by adding attention weights to convolutional neural networks, and the convolutional recurrent neural network (CRNN) is produced by concatenating convolutional neural networks with recurrent neural networks. The author utilized a Twitter dataset with informal language and an adverse drug effects (ADE) dataset obtained by sampling from MEDLINE case reports to evaluate alternative NN architectures. Similar studies related to adverse drug reactions (ADRs) were reported in [9,10]. No doubt there are a plethora of research literature focusing both on adverse drug reaction classification, medical entity recognition and biomedical text summarizing. But a complete system responsible for producing a biomedical text summarizing of adverse drugs is the need in the current scenario.

In this paper, we aim to design a deep-learning-based system that can assist in pharmacovigilance, and biomedical summarization. That means the system not only enables medical researchers to detect, assess, understand and prevent the adverse effects or any other medicine/vaccine-related problems, but also avoid information overload by reducing the length of a medical document while preserving the contents' context and essence. This system is referred to as the intelligent medical information retrieval and summarization system (IMIRSS), which is a solution proposed to help clinicians and medical practitioners in fast and intelligent medical information retrieval. The whole system comprises three main modules, namely the adverse drug event classification (ADEC) module, medical named entity recognition (MNER), and the multi-model text summarization (MMTS) module. In the current study, we design and present the working of the ADEC module to differentiate between ADR and Non_ADR instances.

Social media and publicly available databases are considered to be important sources of information for pharmacovigilance. In pharmacovigilance, at every level of the drug development cycle (DDC), pharmaceutical companies are required to prepare reports called development safety update reports (DSURs) and periodic safety update reports (PSURs) to ensure drug and product safety [11].

The biomedical literature is considered one of the principal information sources that need to be observed in pharmacovigilance [12]. In order to gather data that are important to safety, and in particular, examine the adverse drug reactions (ADRs) connected with a certain drug [13], a vast number of scientific abstracts and publications must be screened and/or read in full. Screening and reading the biomedical literature is a time-consuming task and is of critical importance. It needs to be carried out by qualified readers because it calls for specialized knowledge. Given this, solutions that will help human readers complete this work more quickly and efficiently would be greatly beneficial. There are several studies employing different classification models to classify ADE and Non_ADE instances. The main problem with such systems is the poor performance and lack of supporting components to summarize and recognize the biomedical text literature. Here in this study, we aim to enhance the performance of the classification model to provide an optimal and more appropriate and improved classification inputs for the designing of an intelligent medical text summarization system. Furthermore, the novelty of the proposed model lies in the integration of three modules into an AI-based biomedical entity recognition and text summarization system. To the best of our knowledge, such integration has not been reported in the literature yet.

The intelligent medical information retrieval and summarization system (IMIRSS) proposed in this study is one of the solutions in this direction. In the current study, we are presenting the design ADEC module. This classification module has been designed by employing basic machine learning and deep learning techniques, such as logistic regression (LR), decision tree (DT), and Word2Vec-based text convolutional neural network (TextCNN). Details on each classification technique are provided later in the paper.

In order to design the ADEC module for the IMIRSS project, the ADE-Corpus-V2 [5] corpus is utilized to train the classifier. The ADE-Corpus-V2 comprises two categories of instances, ADE and non-ADE, referred to as adverse drug event and non-adverse drug event, respectively. The whole ADE corpus is distributed within three different files, namely *DRUG-AE.rel, DRUG-DOSE.rel, ADE-NEG.txt*. The description of each sub-corpus (file) is as follows: the $Drug - AE.rel$ presents associations between adverse effects and medications. Drug and dosage relationships are provided by $DRUG - DOSE.rel$. All $ADE - NEG.txt$ sentences in the corpus DO NOT mention any drug-related side effects. The two categories of instances in the corpus correspond to the instances (or say user text) mentioning the adverse effects of a drug and the instances with no drug-related adverse effects information as shown in the examples below.

Example 1. *"Benzocaine-induced methemoglobinemia has been reported in man, dogs, and cats". Here "Benzocaine" is the Drug and "methemoglobinemia" is its Adverse effect.*

Example 2. *"The patient appeared to be homozygous for the dibucaine-resistant gene, having only 15% of normal activity in his serum" Here, it does not contain any drug-related adverse effects.*

This paper's main goal is to demonstrate how the suggested method and algorithm, which take into account both linguistic and structural information from a sentence, may significantly improve the performance of the classifier. The source code for the proposed approach is available for the researchers at (https://github.com/Aioshu/Browser/blob/master/CNN_ADE_VS_Non_ADE.ipynb, accessed on 27 June 2022). The primary contributions of this research are outlined as follows.

- Designing the deep-learning-based adverse drug event classification (ADEC) module for intelligent medical information retrieval and summarization system (IMIRSS) is presented with the working architecture.
- A detailed feature extraction and optimal feature engineering methodology is presented to distinguish between ADE and Non_ADE instances.
- A convolutional neural network (CNN/ConvNet)-based prediction system is proposed, and its working architecture is explained.
- The experimental setup comprising three experiments and hyper-parameter tuning to enhance the classifier performance is presented.
- A rigorous comparative study is conducted, where the results of popular existing systems are compared with our best proposed (ADEC) approach.

The paper's remaining sections are as follows: The literature on text summarization is discussed in Section 2. The working architecture of the proposed IMIRSS is explained in Section 4 along with information on data collection and preprocessing. Furthermore, this module discusses the features used for training different machines and deep-learning-based classification models for designing an adverse drug event classifier. A complete setup of experiments conducted in the proposed study is presented in Section 5. Section 6 discusses the results obtained by the experiments using different classification algorithms. To show the potential of selected features and proposed approach, a comparative analysis with popular existing studies is presented in Section 7. Finally, Section 8 concludes the overall work of the adverse drug event classification (ADEC) module in particular and an intelligent medical information retrieval and summarization system (IMIRSS) in general.

2. Literature Review

With the advancement in information technology (IT), the whole of human life has been changed, including medical and healthcare behaviors. The are several applications of IT in medicine and healthcare, such as hospital information systems (HIS), electronic medical records (EMR), electronic patient records (EPR), medical diagnosis systems, medical image systems, etc. However, health records are usually stored in free text. Several techniques, including data mining, information extraction, fuzzy matching, and keyword filtering, have been employed by researchers to analyze health records and medical images in order to extract and analyze relations.

Researchers have also investigated multiple abstractive and extractive models for medical information extraction and summarization. The sequence-to-sequence models (S2S), text-ranking, and K-nearest neighbor (KNN) algorithms are some examples of such techniques. The authors in study [14] used multiscale convolution with multiple attention vectors to the decoder state to enhance the performance of the S2S model. Another study [15] presented a BERT-based model called BERTSUM for extractive summarization. Similarly, the authors in [2] proposed a sentence-based scoring approach for extractive summarization by combining the word and graph-based ranking techniques. Similarly, a number of abstractive strategies for text summarizing have been examined in previous studies. For example, the research work conducted in [16] created a sequence-to-sequence model based on abstractive summarization. The researchers implemented a pointer–generator model to copy words by pointing in the source texts. Finally, the model training was carried out with coverage of track summary to circumvent repetition. In another study, the author [17] implemented a statistical abstractive summarization framework driven by the abstract meaning representation. The core of the method is a structured prediction algorithm that combines the input's semantic graphs into a single summary semantic graph.

Previously, drug safety surveillance (DSS) was mainly dependent on spontaneous reporting systems (SRS). These systems are populated by suspected ADRs reports collected from healthcare professionals and pharmaceutical companies. The Food and Drug Authority (FDA) has one such system called the adverse event reporting system (AERS). According to recent studies, AERS significantly underestimates the frequency of major adverse drug reactions (ADRs) reported in the system, overreports ADRs that are known to occur, and has missing data, duplicate reports, and unexplained causal relationships. As a result, in recent years, research has shifted to include the use of alternative data sources for ADR detection.

Several NLP techniques have been put forth to identify ADRs in textual data, and using information from electronic health records (EHRs) [18,19] has been their main priority with additional sources such as clinical reports [5]. Electronic health records are more comprehensive to describe a patient's medical history, therapies, ailments, and potential risk factors. However, EHRs have their own difficulties, including their pervasiveness and uncertainty, as well as the identification and quantification of risks and results.

Recent research has focused on merging information from many sources due to the limits of a single source, such as the integration or enrichment of information acquired via spontaneous reporting systems with literature discoveries, electronic health records, chemical similarities, and even user-posted data on social media [20]. ADR detection accuracy has been demonstrated to be greatly increased by merging data from various sources.

Furthermore, one of the main disadvantages of an automated summary system is that it focuses on the text's attributes and ignores the importance of the text from the context point of view. However, the proposed study is focused on the semantic-based features contained in the sentences for abstractive and extractive summarization. Prior research has proposed multiple abstractive and extractive summarization models, but unfortunately, a comprehensive evaluation of these models is lacking in the literature. Additionally, the investigation and benchmarking of such text summarization systems on the text summaries have not been focused on yet. Furthermore, the integration of three modules (adverse drug event classification, medical named entity recognition and multi-model text summarization) into an AI-based biomedical entity recognition and text summarization system has not been reported in the literature yet. Therefore, in this project, one of the objectives is to fill these literature gaps by presenting a detailed assessment of the system implemented for abstractive and extractive text summarization. In this study, we present the IMIRSS framework which consists of three modules stated as the *adverse drug event classification (ADEC)* module, *medical named entity recognition (MNER)*, and the *multi-model text summarization (MMTS)* module. The motivation of this study is to provide intelligent summarization with the blended approach of these modules.

3. Architecture of IMIRSS Framework

The working architecture of the proposed *intelligent medical information retrieval and summarization system* (IMIRSS) is shown in Figure 1. As clearly presented in the figure, the system comprises three main modules, namely the *adverse drug event classification (ADEC)* module, *medical named entity recognition (MNER)*, and the *multi-model text summarization (MMTS)* module. Furthermore, as can be noted from the architecture, initial preparations, such as data collection and pre-processing, are carried out in the first two modules (i.e., ADEC and MMTS). Data collection and pre-processing are separately discussed in the Section 4.1. In this section, we will focus on presenting the working of three main modules of the system. It is to be noted here that in the current study, we are discussing the design and working of module 1 (adverse drug event classification (ADEC)) only. Therefore, other modules will not be discussed in detail here.

In the adverse drug event classification (ADEC) module (module 1), the main focus is to employ different classification models to classify ADE and Non_ADE instances. We employ machine learning as well as deep learning techniques, such as logistic regression (LR), decision trees (DT), and convolutional neural networks (CNN), to run experiments and train ADE classifiers. The ADE-Corpus-V2 [5] corpus is utilized to train the classifier. The ADE dataset consists of 13,981 query points with 2 classes (7000 non-ADE data points and 6981 ADE data points). Data are converted to a machine-understandable representation using different embedding and feature representation techniques, prior to feeding them to the algorithm. In Section 4.1, the detailed information about data collection and preprocessing is provided.

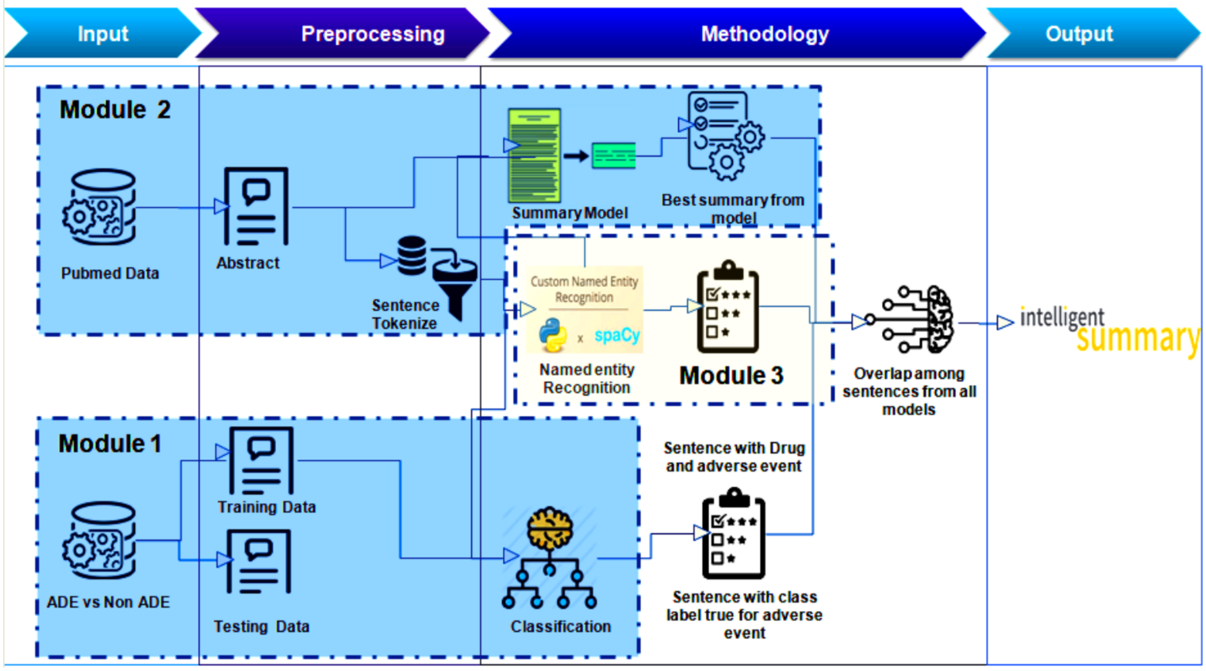

Figure 1. The big picture of the IMIRSS project.

In order to create an optimal feature set, the techniques such as bag of words (BoW), word embedding, such as Word2Vec, and one-hot encoding are utilized. Such techniques convert sentences into a mathematical representation to be easily interpreted by machines. The detailed information about feature engineering is discussed in Section 4.2. After selecting the feature set, the next step is choosing the appropriate learning models. Although there exists a number of ML/DL models for the classification tasks, for the sake of clear understanding, we start with basic categories of models such as linear model (logistic model), tree-based method (decision tree), and neural models (CNN). The complete details of the experimental setup and results obtained by applying models on ADE-Corpus-V2 are presented in Section 5.

Module 2, referred to as the multi-model text summarization (MMTS) module, is mainly responsible for the extractive and abstractive text summarization of medical literature. Several text summarization techniques, such as the centroid-based technique using the k-medoidsalgorithm, state-of-the-art, such as Bio-Bert [21], text rank algorithms [22], and TF-IDF have been implemented. Similarly, in module 3, medical named entity recognition (MNER) is implemented. We employ the Spacy [23] library for the named entity recognition method to efficiently identify drugs and their associated diseases to assure tasks, such as the detection, estimation, assessment, analysis, study, and control of adverse effects of drugs, medicine, or vaccine-related problems.

In module 1, the ADE corpus is split into training and test sets to learn a deep machine model for the classification task. On the other hand, from the same corpus, for the ADR (adverse drug reaction) instances, the information for drugs and diseases is extracted with the regex (regular expression) or/and flash-text (https://flashtext.readthedocs.io/en/latest/, accessed on 11 May 2022) in order to find the offset (start_index, end_index) for each drug and disease. In module 3, the custom NER model is implemented to find the drug and disease in future sentences. The output of module 1, i.e., classification scores, and module 3, i.e., the identification of drug-disease instances, are supplied as input to module 2 in order to perform efficient text summarization, which is one of the main objectives of the proposed IMIRISS project.

Module 2, as a very important part of the IMIRSS, makes use of a different dataset, for example, the PubMed dataset (https://github.com/vgupta123/sumpubmed, accessed on 11 May 2022) used by study [24], consisting of medical abstracts, in order to create intelligent medical summaries for each abstract. For every abstract, two blended approaches are used: in the first approach, we create summaries using several extractive summarization techniques including *text rank, centroid-based, and keyword-based summarizing*, etc., and store the best summarization among all of them using the rouge metric followed by the summary filtering process. In the second approach, the sentence tokenized of NLTK (https://www.nltk.org/, accessed on 11 May 2022) is applied to each abstract and stored in the list. From each sentence list for every abstract, we pass that list of sentences to module 3 and module 1 for recognizing named entities from the instance and identifying adverse drug reaction-related sentences, respectively. In module 3, the associated drugs and diseases in the sentence are detected, filtering out only those sentences with a drug and disease vocabulary. Similarly, in module 1, we obtain the ADR and Non_ADR label for the sentences and filter out only ADR sentences. After the filtering process, from all three modules, we try to see the overlap of the sentences in the abstract and come up with an intelligent summary by using the union or intersection of the sentences.

4. Adverse Drug Event Classification (ADEC)

The primary goal of this study is to design the ADEC module for the IMIRSS framework, and therefore, module 2 (MMTS) and module 3 (MNER) are not discussed in detail here. In the ADEC module, we utilize the ADE-Corpus-V2 [5] corpus to train the classifier. The dataset comprises ADR and Non_ADR samples, divided into train–test sets with a ratio of 80:20. Since we are dealing with text data, it is necessary to convert them into an appropriate form that is supported by ML techniques. Specifically, we initially converted text into a numerical form and then we performed some transformations, including applying regex for filtering, removing stopwords, punctuation, and data cleaning. In the end, we applied the ML techniques with appropriate key performance metrics for evaluating the prediction. Figure 2 presents a flow diagram for the classification (ADEC) module.

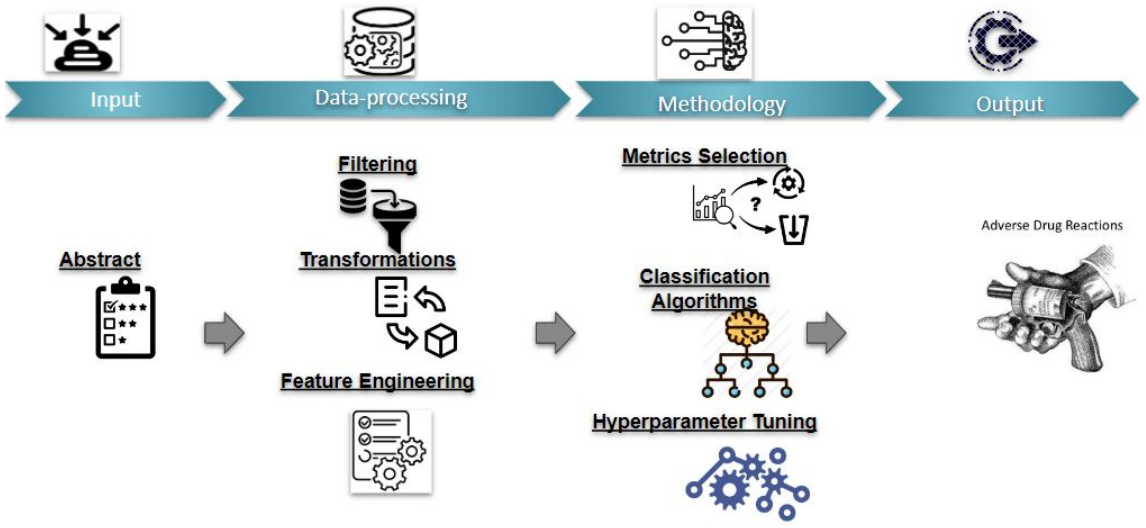

Figure 2. Work flow of classification (ADEC) module of the IMIRSS project.

The algorithm shown in Algorithm 1 is derived from our best-performing technique, i.e., text-based convolutional neural network (TextCNN). We supply a list of sentences as an input parameter and produce output labels as the positive instances (ADR) and negative instances (Non_ADR).

Algorithm 1: Classification of ADR

```
begin
    Input: Sentence_list ($S_L$);
    Output: Sentence Label($S_l$);
            positve_instance (ADR): 1;
            negtive_instance (Non-ADR): 0;
    Initialize Sentence_Embedding_List (SEL);
            SEL := [];
    Initialize Model_Prediction_List (MPL) ;
            MPL := [];
    foreach sentence in Sentence_list do
        Sentence_Vector($S_v$) := Sentence_embedding ($S_i$);
        Sentence_Embedding_List (SEL) :=Sentence_Vector ($S_v$);
        Sentence_Embedding_List (SEL):=Append (Sentence_Vector ($S_v$));
        Model(M) := fit (model(TextCNN));
        Prediction(P) := Model_i(Sentence_Embedding_List (SEL));
        Model_Prediction_List (MPL) := Append (Prediction (P));
    end
end
```

Firstly, we initialize the two empty lists namely Sentence_Embedding_List (SEL) and Model_Prediction_List (MPL) to store the Sentence_embedding (S_i) and prediction (P), respectively. For each sentence from the input, Sentence_list (S_L) is transformed to Sentence_Vector (S_v) and stored in Sentence_Embedding_List (SEL). A TextCNN model is trained using these Sentence_Embedding_List (SEL), and all the prediction by the model is stored in Model_Prediction_List (MPL). The source code for the text-based CNN model is available in our GitHub repository (https://github.com/Aioshu/Browser/blob/master/CNN_ADE_VS_Non_ADE.ipynb, accessed on 11 May 2022) for the researcher to rebuild and reuse the proposed approach.

The three basic tasks performed in the ADEC module are shown in Figure 3. Each step is briefly discussed in the following subsections. In Figure 4, we describe the architecture of the TextCNN model.

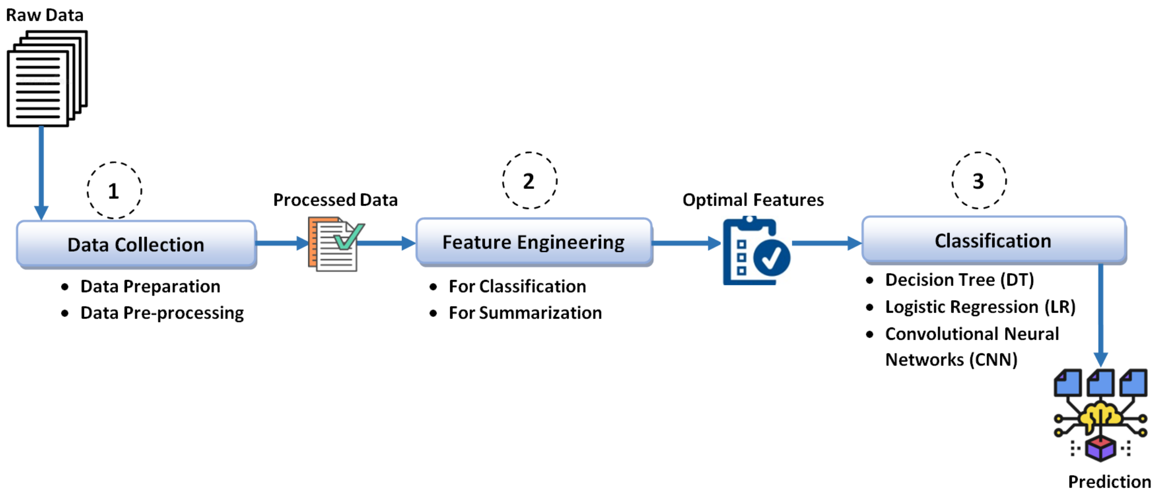

Figure 3. General steps inside adverse drug event classification (ADEC) module.

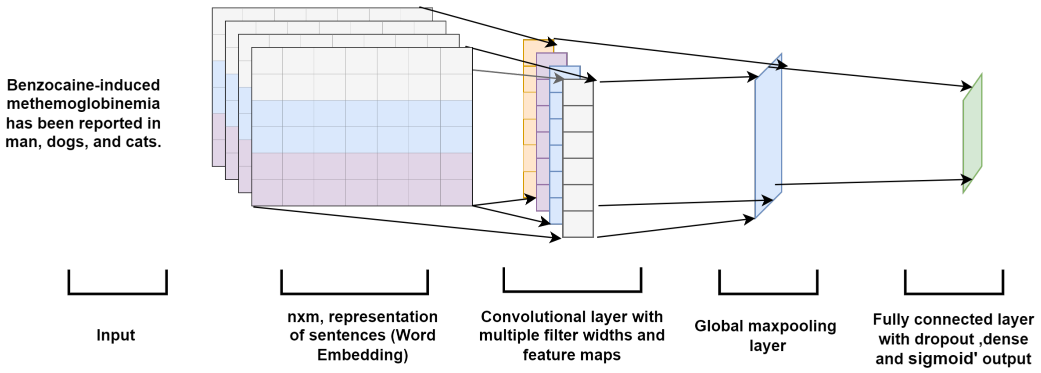
Benzocaine-induced methemoglobinemia has been reported in man, dogs, and cats.
Input
nxm, representation of sentences (Word Embedding)
Convolutional layer with multiple filter widths and feature maps
Global maxpooling layer
Fully connected layer with dropout ,dense and sigmoid' output

Figure 4. Architecture of text-based convolutional neural network (TextCNN).

4.1. Data Collection and Pre-Processing

In this paper, we utilize two datasets for our experiments: one is ADR_v2 corpus (https://github.com/trunghlt/AdverseDrugReaction/tree/master/ADE-Corpus-V2, accessed on 11 May 2022) and the other is collected from Pubmed (https://github.com/vgupta123/sumpubmed, accessed on 11 May 2022). Both datasets needed some preprocessing before feeding them to the algorithm for training.

4.1.1. Data Preparation

The ADE dataset consists of 13,981 query points with 2 classes (7000 non-ADE instances and 6981 ADE instances). The Pubmed dataset consists of 32,684 query points with their reference summaries. From this dataset, 50% of samples are related to our use-case concerning drug and adverse events, and the rest of the instances are not related to the problem we are dealing with; therefore, we discarded them.

4.1.2. Data Preprocessing

In this paper, we are dealing with textual data, so we require text analytical techniques which is natural language processing (NLP). The ultimate objective of NLP is to understand, read, decipher, and make sense of human languages in a manner that is informative. For the NLP related to this study, we employed the NLTK toolkit (https://www.nltk.org/, accessed on 11 May 2022), which comprises text-processing modules for tasks, such as word/sentence tokenization, parsing, lemmatizing, stopwords, stemming, etc.

The primary step in NLP projects is pre-processing. It includes pre-processing, such as *"Removing punctuations" (. , ! $( ) % @)*, *"Removing URLs"*, *"Removing Stop words"*, *"Lower casing"*, *"Tokenization"*, *"Stemming"*, *"Lemmatization"*. Performing the above steps will help to eliminate meaningless tokens and create dense vectors. Initially, the data are pre-processed by converting tokens (words) to lowercase, and removing all unambiguous symbols and stopwords. Lowercasing is one of the most common pre-processing task, where each character in the each letter in the test is changed into lower case. Stopwords and punctuation are common words from the text and carry less or no information; therefore, they are useless for analysis, and unambiguous symbols are misleading while employing a model. Stopwords consist of words such as i, i'm, me, myself, about, themselves, we, ain't, our, ours, and, you've, etc. Stemming is considered another text standardization process where the words are transformed to their root form, for example, *"crazy → crazi"*, *"available → avail"*, *"entry → entri"*, *"early → earli"*. Lemmatization is similar to stemming but ensures that context is preserved. Lemmatization employs a lexicon to check a word's definition while reducing it, for example, *"goose → goose"*,*"geese → goose"*. After the preprocessing tasks, the final data are converted to mathematical form employing BoW, TF-IDF, and word-embedding techniques.

4.2. Feature Engineering and Representation

In this section, we focus on the features which are important for both classification as well as text summarization tasks. Since both models take inputs in different forms, thus we convert the input data (features) to the form that each model supports. In order to find the optimal features and improve the model performance, we employ the attribute individual power strategy in which several techniques, such as TF-IDF, bag of words (BoW), n-grams, etc., are used to know the potential of selected features.

4.2.1. Feature Representation for Classification

In the proposed work, for the ADE corpus, we take every sentence labeled (ADE and non-ADE) in the dataset as mentioned previously. However, before feeding the data to a model, they need to be in the machine-intelligible form. Therefore, we employ feature representation techniques, such as term frequency–inverse frequency (TF-IDF), bag of words (BoW), word embedding (https://radimrehurek.com/gensim/models/word2vec.html, accessed on 11 May 2022), 1-hot-vector encoding (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html, accessed on 11 May 2022), etc. These feature representation techniques depend on the term set and their occurrences in the corpora, and, therefore, are unable to identify the structure of a sentence in a paragraph.

Word embedding is another technique for representing features, where all the unique words in the corpus are grouped in vectors in which semantically similar words will share the same vector space.

For the word embedding vector of tokens in the sentences, we have utilized the Word2Vec model. The Word2Vec model contains more than 3 million English words with each word represented by 300*1-dimensional word vectors. To extract word vectors using the Word2Vec model, the Gensim library (https://pypi.org/project/gensim/, accessed on 11 May 2022) is employed, but for the medical literature, there are some scientific terms that are not well captured by word embeddings.

On the other hand, with one-hot encoding, a binary vector is associated with every token, thus representing all the distinct words in different way. Hence, we consider one-hot encoding representation as a better choice for presenting sentences to our model.

4.2.2. Feature Representation for Text Summarization

For text summarization, the data are collected from PubMed which contains medical abstracts. For each abstract, we generate the summary of the abstracts using the frequency feature methods and Word2Vec embedding methods to create features for the text summarization module2 (MMTS).

In the case of frequency-based summary, we use TF-IDF features. The TF-IDF score is a parameter for analyzing the importance of sentences in an abstract.

In formal ways of calculating TF-IDF, as follows,

$$T_f - ID_f :\ = (T_f * ID_f) \tag{1}$$

both the terms T_f and ID_f are calculated in separate equations as follows:

$$T_f :\ = \frac{(\#repetitions\ of\ a\ given\ sentences\ w\ in\ an\ abstract\ (d)}{(\#\ unique\ terms\ in\ an\ abstract\ (nd))} \tag{2}$$

$$ID_f :\ = \frac{log(\#\ total\ abstract\ (D)}{(\#\ of\ abstract\ containing\ sentences\ (nd \in D))} \tag{3}$$

The TF-IDF (term frequency–inverse document frequency) statistic examines the relation of a word to a document. This is accomplished by multiplying two metrics: the frequency of a word appearing in a document, and the word's inverse document frequency over a collection of documents. The rationale behind the TF-IDF approach is as follows:

- A word that frequently appears in a text has a strong relationship with that document, which increases the likelihood that the document is about that particular word or term.
- A word that appears in several documents routinely may deceive in extracting the document of interest in the collection. This implies that the word is related either to all documents or none. It will not help us in finding out a document or a small subset of documents from the document set.

Thus, TF-IDF is a value computed from the words in every document of the document set (dataset). The TF-IDF value for every word increases as per the frequency of the specific word in a document and decreases with a word's appearance in other documents. In the case of Word2Vec-embedding, each sentence from the abstract is converted into vectors. These vectors are later used for training the k-medoids algorithmsfor calculating the centroids and generating extractive summaries of the abstracts. It should be again noted here that this study is focused on designing module 1 (AERC) of the IMIRSS project. Therefore, we are not going into more detail about modules 2 and 3.

4.3. Classification Models

This section will briefly discuss the classification models which have been employed during the experiments, in ADEC module, to classify ADE and Non_ADE instances. The models were trained on the ADE-Corpus-V2 [5] corpus, which contains two types of instances: ADE (adverse drug event) and non-ADE (non-adverse drug event). The dataset has 13,981 query points in the ratio of 7000:6981 for non-ADE and ADE samples, respectively. We split our dataset into train–test using the 80:20 principle with a random state for uniformity. Since we are dealing with text data, and therefore the chance of the presence of unwanted content as well, it is necessary to clean unstructured and raw data before converting them into numerical form. To achieve this, apart from NLTK pre-processing functionalities, we also applied some transformations, such as regex for filtering stopwords, unnecessary symbols, etc. Prior to feeding the data to the algorithm, they are converted to a machine-understandable representation using various embedding and feature representation techniques. The comprehensive information regarding feature representation is provided in Section 4.2. Experiments are conducted using deep/machine learning techniques, such as logistic regression (LR), decision trees (DT), and convolutional neural networks (CNN), which are described briefly in the following subsections.

4.3.1. The Logistic Regression

The logistic regression (LR), also known as a generalized linear model, enables the probabilistic modeling of binary variables. The LR makes use of the logit function, also referred to as log-odds or the logarithm of odds. The odds ratio is considered the proportion of the odds of event *a* when event *b* is present to the odds of event *a* when event *b* is absent.

Equation (4) presents the mathematical explanation of the logistic regression (LR).

$$\begin{aligned} ln(P/(1-P)) &= Beta_0 + Beta_1 * X_1 + \ldots + B_k * K_k \\ P &= 1/(1 + exp^{-}(\theta_1 + \theta_2 x)) \\ \sigma(z) &= 1/(1 + exp(\theta_1 + \theta_2 x)), where\ z = \theta^T x \\ \theta^T x &= \theta_1 + \theta_2 x + ... + \theta_2 m \end{aligned} \tag{4}$$

where

- The probability for event Y to occur is $P : P(Y = 1)$.
- $P/(1 - P)$ denotes the odds ratio of events.
- θ denotes the parameters having length m.

The probabilities between 0 and 1 are calculated with the Logit function, thus LR is a non-linear function and looks like the S-function as shown in Figure 5. We use logistic regression because linear regression is not appropriate for modeling binary outcomes because of two reasons. First, a linear model makes unbounded continuous predictions. In binary classification, we intend to find the probability of an outcome occurring; therefore, we are interested in predictions bounded between 0 and 1. Second, the linear models violate the assumption of normal residuals in binary prediction outcomes, thus, distorting inferences made on regression coefficients. With binary data (as with our case), the variance is a function of the mean, and is not constant as the mean changes. This violates one of the standard linear regression assumptions that the variance of the residual errors is constant.

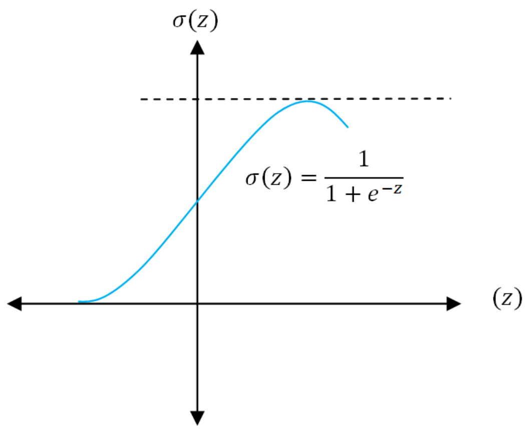

Figure 5. Logistic regression.

4.3.2. Decision Tree

A decision tree (DT) is a tree-based method that can be used for classification tasks. The DT uses a tree structure to show the predictions obtained from a series of splits based on features. It begins with a root node and stops with a decision at leaves. There are a few terminologies: *root nodes*—exist at the initial level of a decision tree and based on this node, the splitting process starts; *decision nodes*—root nodes are further divided into decision nodes; *leaf nodes*—based on these nodes, the splitting stops; and sub-tree—sub-section of the decision tree. Finally, pruning is referred to as the slicing of some nodes from the decision tree in order to avoid over-fitting. The pictorial representation of decision tree with its components is shown in the Figure 6.

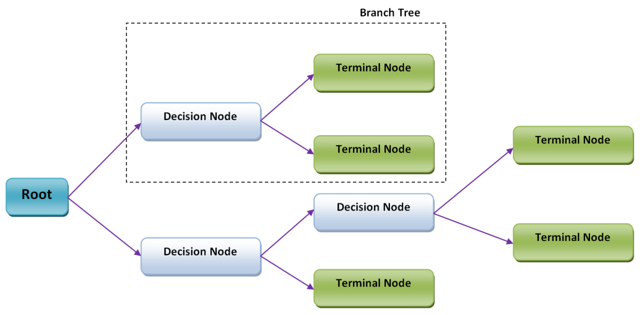

Figure 6. Decision tree.

Entropy is measured, based on which the split happens; it is referred to as the randomness in the dataset under consideration.

$$E(S) = -p_{(+)} log p_{(+)} - p_{(-)} log p_{(-)} \quad (5)$$

Equation (5) for entropy is explained as follows: where $p_{(+)}$ represents the positive probability of a class, $p_{(-)}$ represents the negative probability of a class, and S denotes the fragment of the training instances.

Entropy gives the impurity of a node, but it is always unsure if the entropy of that specific node has decreased or not.

In order to avoid this, we apply the metric called "information gain" (IG) to calculate the decrease in the entropy.

$$Information\_Gain(IG) = E(Y) - E(Y|X) \tag{6}$$

Equation (6) for information gain (IG) is described as follows: where $IG(Y, X)$ refers to information gain for dataset Y for a random variable X. Here, $E(Y)$ indicates the entropy for the dataset before any change, whereas $E(Y|X)$ indicates the conditional entropy for the dataset Y given the variable X. Based on some features, IG calculates the decreases in the randomness of the whole dataset under consideration.

4.3.3. Text-Based Convolutional Neural Network (TextCNN) Architectures

Convolutional neural network (CNN) is based on an artificial neural network, initially designed for image recognition and specifically implemented to analyze pixel data. CNN convolutional layers are different than general neural networks. In order to classify images, the network iterates through each vector and each matrix dimension, making CNN more resilient to matrix data. Convolutional layers are a basic instrument for CNN to perform modeling while detecting more complicated features because they hold many functionalities, including recognizing edges, corners, and multiple textures. The dimension of the convolutional layer expands with the expansion of features. We must use a convolution layer with a single dimension because our data are in the form of text and can be viewed as sequential in a one-dimensional matrix. The CNN architecture is much the same, but the convolution layers' data type and dimension are different. In order for the implementation of text-based CNN (TextCNN), we need a one-dimensional convolutional network and a word-embedding layer. The general architecture of the TextCNN model is shown in Figure 7. Instead of a pixel matrix, text as a sequence is fed to the TextCNN, and the embeddings matrix is passed to the embedding layer of TextCNN.

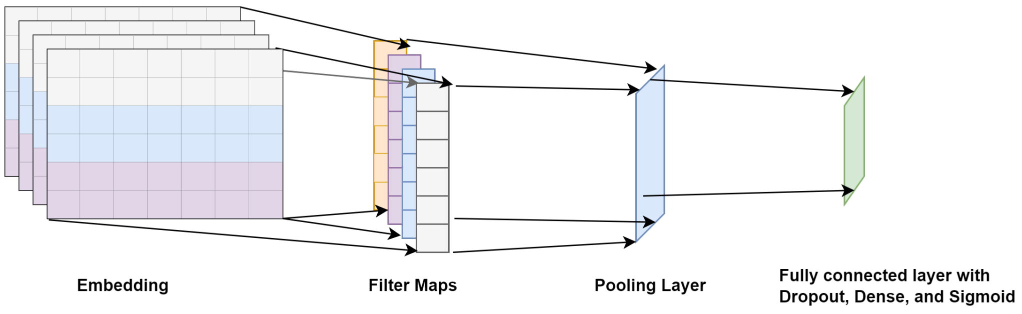

Figure 7. General architecture for convolutional neural network.

4.3.4. Majority Voting Classifier

The majority voting classifier model is the type that combines the predictions of multiple models to create an ensemble prediction. We have two types of voting settings, viz, soft voting and hard voting. The soft voting setting instructs the model to average the probabilities generated by each of the models provided to it. If some models perform more reliably than the others, we might apply weighting to the voting calculation to favor those more. On the contrary, a hard voting setting demands cherry-picking the prediction with the highest number of votes. Since we utilize three models (LR, DT, and CNN) in the ADEC module and the best practices, we will be using a soft voting mechanism so the predictions are weighted by the classifier's importance and summed. The target label with the greatest sum of weighted probabilities wins the vote.

5. Experimental Setup

The prime objective of this paper is to evaluate the performance of abstractive and extractive text summarization models to detect adverse drug events (ADE). Furthermore, the paper also performs the classification of adverse drug effects, which will help pharmaceutical companies and health professionals in making medications (drugs) safer and advocating for patients' safety.

As mentioned before, we used the adverse drug event (ADE) dataset [5] with more than 13k samples. The whole dataset is divided into two parts in the ratio of 80:20 for training and testing, respectively. Furthermore, the training part is further divided into an 80:20 ratio for training and cross-validation, respectively. It is always considered good practice to split the data into three (training, validation, and test) subsets rather than just two (training, validation/Test) sets because this way, one can prevent the trained model from overfitting and evaluating it more effectively. The validation subset is mostly utilized to describe the evaluation of the model when tuning hyperparameters and sometimes during data preparation, and the test set is predominately used to evaluate a final tuned model while comparing to other existing approaches or models.

Figure 8 presents a clear visualization of the experimental setup established for the proposed study.

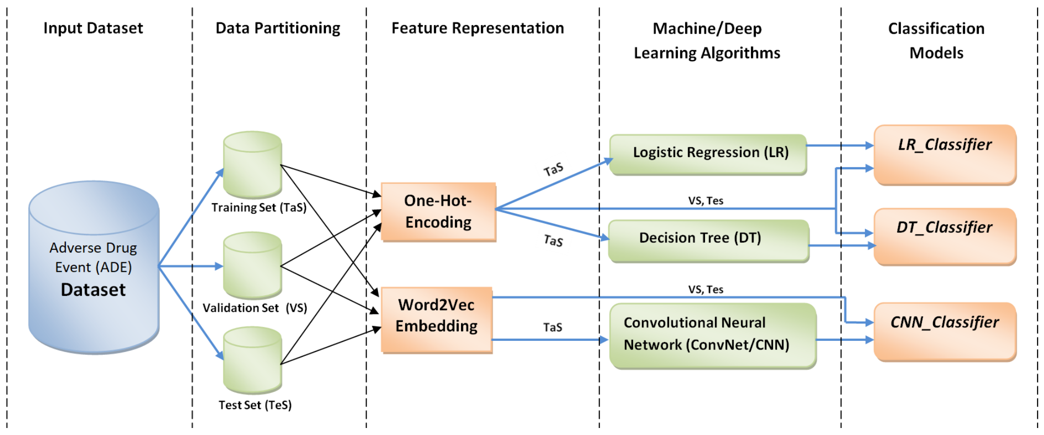

Figure 8. Experimental setup for the proposed study.

In order to evaluate the performance of the proposed model, we use commonly known performance measures such as accuracy, precision, and recall. Based on the confusion matrix given in Table 1, the formula for calculating each of the metrics is as follows:

$$Accuracy(A) = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{8}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{9}$$

$$F1 - score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{10}$$

Table 1. Confusion Matrix.

		Actual	
		Positive	**Negative**
Predicted	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

6. Results and Discussion

As mentioned, we applied classical machine learning and the deep learning algorithm to the ADE dataset. For the LR model, the whole dataset after partitioning into training, validation, and test sets, we yielded 8844, 2765, and 2212 data points, respectively.

The learning rate, denoted by the symbol α, is a hyperparameter used to govern the pace at which an algorithm updates or learns the values of a parameter estimate. The rate of learning or speed at which the model learns is controlled by the hyperparameter. It controls how much allotted error is utilized to update the model's weights every time it is revised, for example, at the completion of every batch of training instances. A desirable learning rate is low enough for the model to converge on something useful, while being high enough to train in a reasonable length of time. Smaller learning rates necessitate more training epochs because of the fewer changes. On the other hand, larger learning rates result in faster changes. Moreover, larger learning rates frequently result in a sub-optimal final set of weights.

As shown in Figure 9, we get different values of alpha in a list [0.00001, 0.0001, 0.001, 0.01, 0.1, 1], we obtained minimum log loss: 0.482 at alpha = (0.00001) for logistic regression model. Therefore, we use alpha = 0.00001 to record the maximum accuracy. The performance of Logistic Regression (LR) and Decision Tree (DT) measured in terms of precision, recall and accuracy is shown in the Table 2.

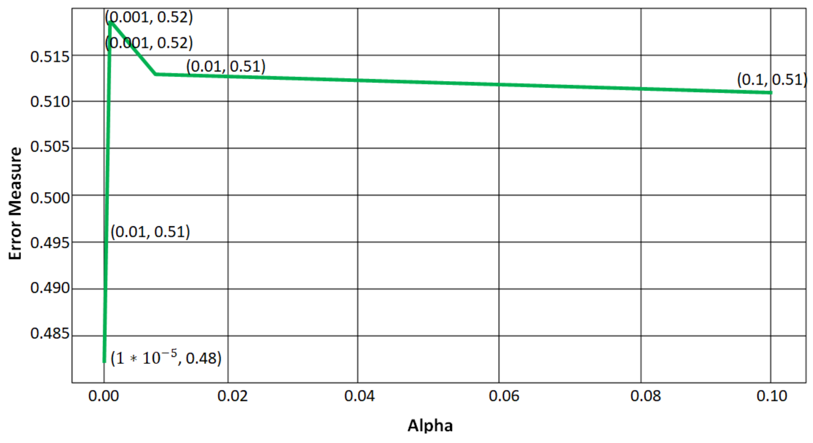

Figure 9. Cross Validation Error.

Table 2. Performance of several supervised machine learning techniques on ADE Dataset.

Technique	Precision	Recall	Accuracy
Logistic Regression (LR)	0.80	0.92	0.85
Decision Tree (DT)	0.79	0.76	0.77

It is well noted that a basic classification model cannot perform effectively in complicated datasets where the creation of vocabulary increases the size of the matrices. In this case, a state-of-the-art model, such as CNN, is utilized to interpret relationships between tokens in a sentence.

Even though CNN is a type of neural network, it only varies from other neural networks due to its convolutional layer. CNN examines each pixel matrix's corner, each vector, and each dimension while classifying images. It becomes more resilient to matrix-based data by operating with all of its capabilities. Based on this analogy, we applied CNN to textual data, as it is like sequential data in time series, and we can consider it as a matrix with one dimension. We employed the Word2Vec model which produces an embedding vector as an input for our CNN model (CNN Layer). The working architecture of our CNN model is presented in Figure 4.

After 120 epochs, we obtain the train accuracy of 98% and the test accuracy of 89% as presented in Figure 10. Furthermore, Figure 10 also presents the training loss and validation loss of 0.232 and 0.11, respectively, for the proposed CNN model. The confusion matrix that we obtained for the CNN model is shown in Figure 11. It should be noted that we only presented the confusion matrix of our best-performed model, which is the CNN model.

For the CNN model, we have almost 13,821 instances, out of which 13,129 instances are used for training, and 692 are utilized to validate the model's performance.

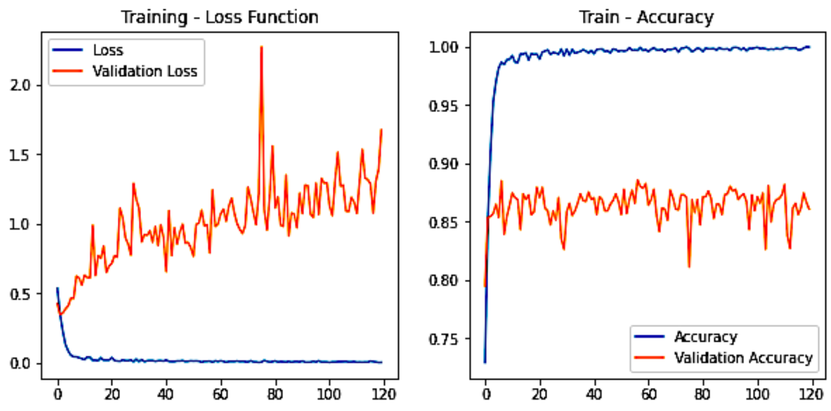

Figure 10. Train loss and train accuracy.

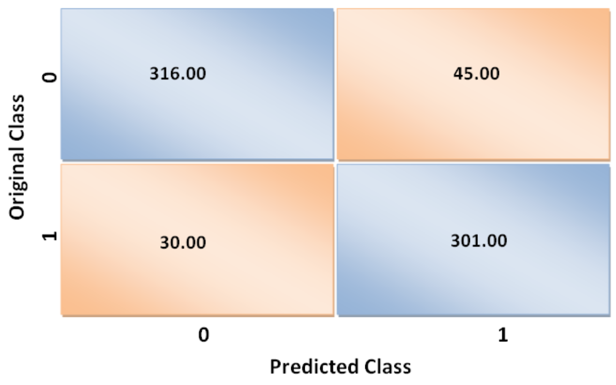

Figure 11. Confusion matrix for text-based convolutional neural network (TextCNN).

From the confusion matrix, we also recorded the values for the other performance measures, such F_1-score, recall, and precision for the implemented CNN model, as presented in Table 3. As clearly noted in the table, we have shown the accuracy, precision, and recall for both positive (ADR) and negative (Non_ADR) classes represented with 0 and 1, respectively.

Table 3. Metric: classification report proposed CNN model.

	Precision	**Recall**	**F1**	**Support**
0	0.91	0.88	0.89	361
1	0.87	0.91	0.89	361
accuracy	0.89	0.89	0.89	692
macro avg	0.89	0.89	0.89	692
weighted avg	0.89	0.89	0.89	692

As it can be noted from the above table, the F1-score for both positive and negative classes is 89%. The F1-score is considered the harmonic mean between values of precision and recall. The formal way of calculating the F1-score is presented in Equation (10). Support is the count of instances of the actual response that lie in that particular class. In our case, the count of support instances from both positive and negative classes is 361, as shown in Table 3. For the positive class (ADR), we obtained precision, recall, and F1-measure as 91%, 88%, and 89%, respectively, whereas for negative class (Non_ADR), the recorded values for precision, recall, and F1-measure are 87%, 91%, and 89%, respectively.

In order to gain an understanding of the performance of our trained model, we show a few examples from our dataset where the model classifies correctly and incorrectly the instances in Table 4 and Table 5, respectively.

Table 4. Examples of correctly classified ADR/non-ADR dataset.

ID	Sentence	Label True	Label Predict
1	In vitro rechallenge of the patient's lymphocytes with cytochrome P-450 generated metabolites of phenobarbital showed extensive cytotoxicity compared to control.	Non_ADR	Non_ADR
2	PURPOSE: The case of a patient who developed aseptic meningitis, hemolytic anemia, hepatitis, and orthostatic hypotension simultaneously during treatment with trimethoprim-sulfamethoxazole is described.	ADR	ADR
3	The necessity of placing such grafts behind the ureters is reiterated and the routine performance of preoperative and postoperative excretory urography is suggested.	Non_ADR	Non_ADR
4	An 83-year-old man receiving glipizide 10 mg bid developed symptomatic hypoglycemia within three days of adding trimethoprim/sulfamethoxazole (TMP/SMX) to his regimen.	ADR	ADR
5	A patient is described who developed a poorly differentiated sarcoma after cyclophosphamide was used to treat his rheumatoid arthritis.	ADR	ADR
6	The patient appeared to be homozygous for the dibucaine resistant gene, having only 15% of normal activity in his serum.	Non_ADR	Non_ADR

Table 5. Examples of misclassified ADR/non-ADR dataset.

ID	Sentence	Label True	Label Predict
1	Pharmacokinetically induced benzodiazepine withdrawal.	Non_ADR	ADR
2	Benzocaine-induced methemoglobinemia was reported in men, dogs, and cats.	ADR	Non_ADR
3	Transient visual anomalies associated with drug treatment for spastic colon.	ADR	Non_ADR
4	A patient with generalized MG was effectively managed with MM but developed CNS lymphoma after 3 years of treatment.	ADR	Non_ADR

Table 5 shows misclassified data points. Since we used the CNN model which is a black-box model, it is very hard to obtain the interpretability from the model. Since we have the ground truth labels, we easily find a pattern of incorrectly classified data points. As shown in Table 4, correctly classified samples have a greater length as compared to misclassified data points in Table 5. From the medical domain point of view, in the first instance, ***"Pharmacokinetically induced benzodiazepine withdrawal"***, the model misclassifies it because it gets the semantics of 'induced by the drug'. **Pharmacokinetically** means the process by which a drug is absorbed, distributed, metabolized, and eliminated by the body. **Benzodiazepines** are drugs that relieve anxiety. Since we are using word embedding, it finds the word 'induced' which is a proper English word and a drug name in it.Therefore, the model gets confused and misclassifies the point.

7. Comparative Analysis

This section compares the proposed classification module of the IMIRSS project with a well-known study [7] in the literature to present the performance of a deep-learning-based classification model. The experiment findings demonstrate that the TF-IDF and one-hot encoding techniques are less suitable for constructing our IMIRSS systems since deep learning techniques (TextCNN), which are represented using word embedding as features, are much more suitable for predicting the outcome. We execute sets of experiments as our benchmark and compare them to the models that the authors have proposed in their work. In Ref. [7], the author employed several machines and deep-learning-based models, such as term-matching (TM), maximum-entropy classifier with n-grams and TF-IDF weightings (ME-TFIDF), maximum-entropy classifier with n-grams and NB log-count ratio (ME-NBLCR), and maximum-entropy classifier with mean word embedding (ME-WE). Furthermore, the authors employed CNN-based approaches, such as the recurrent variant of CNN (RCNN), convolutional recurrent neural network (CRNN), and CNN with attention (CNNA), and showed that out of all approaches, the CNN-based system is the best performing. The authors evaluated the performance of the trained models based on the F1-score, precision, and recall of the positive class (instances with the information of adverse drug reactions) with the baseline models as shown in Table 6. It can be observed from the table that the ME-NBLCR approach outperforms ME-TFIDF on the ADE dataset, with an F1 score of 0.84 as compared to ME-TFIDF (0.80). Furthermore, MaxEnt from aggregated word embedding (ME-WE) (0.57) performs more poorly than both the ME-TFIDF (0.80) and ME-NBLCR (0.84) methods. CNN with an F1-score of 0.87 outperforms alternative neural network architectures (CRNN (0.84), RCNN (0.83), CNNA (0.83)), with a performance improvement of 3–4% in the F1-score. To make our work more reliable, we are comparing the proposed approach with another study [25] toward drug-adverse event extraction using machine learning. The authors in this study reported a precision of 0.85 and recall of 0.84 for causal sentence classification and a precision of 0.72 and recall of 0.77 for suspect drug identification on reliable test data, while the proposed CNN-based approach obtained precision, recall, and F1 measure of 0.87, 0.91, and 0.89, respectively.

Table 6. Performance of existing models on ADE dataset.

Technique/Approach	Precision (P)	Recall (R)	F1-Score
ME (Maximum-Entropy)-TFIFDF	0.74	0.86	0.80
ME (Maximum-Entropy)-NBLCR (NB Log Count Ratio)	0.91	0.79	0.84
ME-WE (Maximum-Entropy classifier with mean word embeddings)	0.48	0.70	0.57
CNN	0.85	0.89	0.87
CRNN (Convolutional Recurrent Neural Network)	0.82	0.86	0.84
RCNN (Recurrent Convolutional Neural Network)	0.81	0.89	0.83
CNNA (Convolutional Neural Network with Attention)	0.82	0.84	0.83
Proposed Approach	↓	↓	↓
DT-OneHotEncodding	**0.79**	**0.76**	**0.77**
LR-OneHotEncodding	**0.80**	**0.92**	**0.85**
CNN-Embedding	**0.87**	**0.91**	**0.89**

Furthermore, the authors aimed to provide a machine learning (ML) and natural language processing (NLP) based solution for extracting suspect drugs and adverse events, which is different than our study in which our main goal is to design an intelligent medical information retrieval and summarization system (IMIRSS). Adverse drug event classification (ADEC) is one of the modules of the proposed system.

In order to evaluate the performance of the proposed model, a similar CNN-based system with a Word2Vec embedding approach is implemented. Similar to the author's study [7], we employed techniques on the same corpus. In an initial experiment, the words from the sentences were extracted using the bag-of-words (BOW) technique, i.e., one-hot encoder. As a baseline model, we created the logistic regression, as it is more interpretable and works well with the textual content. The values recorded for this baseline model for performance matrices, such as F1-score, precision, and recall, were 0.85, 0.80, and 0.92, respectively. In the second experiment, we implemented tree-based methods in case of categorical data with no data transformation required, but we observed a drop in F1-score by 8%, precision by 1%, recall by 16% from the baseline model. Based on machine learning, a deep learning approach in the third experiment was used, where we implemented text-based convolutional neural network with Word2Vec embeddings and minor changes in the values of the parameters, such as p-value [26] and changing the max-pooling layer to the global max-pooling layer [27].Luckily, we obtained comparatively higher values for F1-score, precision, and recall as 0.89, 0.87, and 0.91, respectively. Therefore, in comparison to the author's best model in Table 6, our text-based CNN with Word2Vec embeddings outperformed with 2% in precision, recall, and F1-score, as shown in Table. Thus, it is to be concluded here that the text-specific neural network models combined with the Word2Vec embeddings techniques perform better than TF-IDF-based maximum entropy and maximum entropy classifier with n-grams and NB log-count ratio (ME-NBLCR) for the text classification task. it is also noted that the proposed CNN with a global max-pooling layer suppressed the variants of CNN (CRNN, RCNN, and CNNA) in the existing literature.

8. Conclusions and Future Scope

The exponential growth in healthcare records, stored in large databases need professional readers to retrieve the information timely, thus making the process lengthy, and time-consuming for clinicians and medical practitioners, therefore, demands, new, fast, and intelligent medical information retrieval and presentation methods.

This study is a part of the project which aims to design an AI-based *medical information retrieval and summarization system referred to as IMIRSS*. The whole system comprises three main modules: the *adverse drug event classification (ADEC)* module, *medical named entity recognition (MNER)*, and the *multi-model text summarization (MMTS)* module. In the current study, we present the design ADEC module. This classification module is designed by employing basic machine learning as well as deep learning techniques, such as logistic regression (LR), decision tree (DT), and text-based convolutional neural network (TextCNN). Models such as Word2Vec and TF-IDF are used to derive features from the dataset. All the trained models have produced promising results. TextCNN with an accuracy of 89% performs better than the conventional machine learning approaches LR and DT with accuracies of 85% and 77%, respectively. Furthermore, the best performing TextCNN (with global maxpooling layer with activation function ReLU and dropout layer with p-value as 0.1 and 0.2) outperformed the existing adverse drug event approaches on the performance metrics such as precision, recall, and F1-score with 87%, 91%, and 89%, respectively.

To improve the performance of the overall IMIRSS system for efficient information retrieval and summarization, we employed an ensemble strategy for the outputs produced by each classification model inside the ADEC module. The predictions of selected base models were combined in order to improve the robustness of the single model. The future scope in this direction involves designing the remaining two modules, viz *medical named entity recognition (MNER)*, and *multi-model text summarization (MMTS)* to design an efficient *medical information retrieval and summarization system*. The main problem with existing systems is the poor performance and lack of supporting components to summarize and recognize the biomedical text literature. As discussed in Section 7, the proposed classification model achieved better results than the existing methods, thus providing optimal and more appropriate and improved classification inputs for the designing of an intelligent medical text summarization system.

Furthermore, the novelty of the proposed model lies in the integration of three modules into an AI-based biomedical entity recognition and text summarization system. To the best of our knowledge, such integration has not been reported in the literature yet.

The proposed work is useful for society in several ways. From the ADE classification point of view, the proposed system assists healthcare societies worldwide to identify new, rare and serious adverse drug reactions and to evaluate drug risk factors for preventing future adverse drug reactions (ADRs). Apart from patient safety, it also helps to protect healthcare organizations from costly liability claims and financial losses caused by reduced reimbursement for the treatment of preventable conditions acquired while in hospital care.

The IMIRSS system as a intelligent medical text summarization system helps medical practitioners by reducing the time to access important information from countless documents, thus providing quick and proved medical consultations. Furthermore, the biomedical text summarization system helps biomedical information seekers avoid information overload by reducing the length of a document while preserving the contents' essence.

The strategy in this study is to start simply by using machine learning algorithms that are more interpretable rather than going to black-box models directly. However, surely, in the future, we would like to evaluate the transfer learning models, where we reuse a pre-trained model as the starting point for a model on a new task. Furthermore, from the ADEC module point of view, apart from the Word2Vec and TF-IDF representations, we can also explore pre-trained models, such as GloVe, and FastText, to evaluate the performance of the classification model.

Author Contributions: Conceptualization, A.R. and M.A.W.; methodology, A.R. and M.A.W.; software, A.R.; validation, A.R., M.A.W., A.S.I. and Z.K.; formal analysis, M.A.W., A.S.I., Z.K., M.E. and S.M.D.; investigation, A.R. and M.A.W.; resources, M.A.W., M.E.; data curation, A.R. and M.A.W.; writing—original draft preparation, A.R. and M.A.W.; writing—review and editing, M.A.W., A.S.I. and Z.K., M.E. and S.M.D.; visualization, A.R. and M.A.W.; supervision, A.S.I., Z.K., M.E. and S.M.D.; project administration, A.S.I., Z.K., M.E. and S.M.D.; funding acquisition, M.A.W., M.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is supported and funded by the EIAS, Data Science and Blockchain Laboratory, Prince Sultan University, Riyadh, Saudi Arabia. The APC was also funded by Prince Sultan University.

Data Availability Statement: Not applicable.

Acknowledgments: This research work is supported by the EIAS, Data Science and Blockchain Laboratory, Prince Sultan University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Allahyari, M.; Pouriyeh, S.; Assefi, M.; Safaei, S.; Trippe, E.D.; Gutierrez, J.B.; Kochut, K. Text summarization techniques: A brief survey. *arXiv* **2017**, arXiv:1707.02268.
2. Chen, F.; Han, K.; Chen, G. An approach to sentence-selection-based text summarization. In Proceedings of the 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02, Beijing, China, 28–31 October 2002; Volume 1, pp. 489–493.
3. Moratanch, N.; Chitrakala, S. A survey on abstractive text summarization. In Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 18–19 March 2016; pp. 1–7.
4. Vines, T.; Faunce, T.A. Assessing the Safety and Cost-Effectiveness of Early Nanodrugs. 2009. Available online: https://pubmed.ncbi.nlm.nih.gov/19554862/ (accessed on 11 May 2022).
5. Gurulingappa, H.; Rajput, A.M.; Roberts, A.; Fluck, J.; Hofmann-Apitius, M.; Toldo, L. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *J. Biomed. Inform.* **2012**, *45*, 885–892. [CrossRef]
6. Shakib, S.; Caughey, G.E.; Fok, J.S.; Smith, W.B. Adverse drug reaction classification by health professionals: Appropriate discrimination between allergy and intolerance? *Clin. Transl. Allergy* **2019**, *9*, 18. [CrossRef] [PubMed]
7. Huynh, T.; He, Y.; Willis, A.; Rüger, S. Adverse drug reaction classification with deep neural networks. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016.
8. Miranda, D.S. Automated detection of adverse drug reactions in the biomedical literature using convolutional neural networks and biomedical word embeddings. *arXiv* **2018**, arXiv:1804.09148.

9. Kaufman, G. Adverse drug reactions: Classification, susceptibility and reporting. *Nurs. Stand.* **2016**, *30*, 1–10. [CrossRef] [PubMed]
10. Sarker, A.; Gonzalez, G. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *J. Biomed. Inform.* **2015**, *53*, 196–207. [CrossRef] [PubMed]
11. Krishnamurthy, A.C.Y.; Dhanasekaran, J.; Natarajan, A. A succinct medical safety: Periodic safety update reports. *Int. J. Basic Clin. Pharmacol.* **2017**, *6*, 1545. [CrossRef]
12. Pontes, H.; Clément, M.; Rollason, V. Safety signal detection: The relevance of literature review. *Drug Saf.* **2014**, *37*, 471–479. [CrossRef]
13. Alhuzali, H.; Ananiadou, S. Improving classification of adverse drug reactions through using sentiment analysis and transfer learning. In Proceedings of the 18th BioNLP Workshop and Shared Task, Florence, Italy, 1 August 2019; pp. 339–347.
14. Tjandra, A.; Sakti, S.; Nakamura, S. Multi-scale alignment and contextual history for attention mechanism in sequence-to-sequence model. In Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 18–21 December 2018; pp. 648–655.
15. Liu, Y. Fine-tune BERT for extractive summarization. *arXiv* **2019**, arXiv:1903.10318.
16. Nallapati, R.; Zhou, B.; Gulcehre, C.; Xiang, B. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv* **2016**, arXiv:1602.06023.
17. Liu, F.; Flanigan, J.; Thomson, S.; Sadeh, N.; Smith, N.A. Toward abstractive summarization using semantic representations. *arXiv* **2018**, arXiv:1805.10399.
18. Friedman, C. Discovering novel adverse drug events using natural language processing and mining of the electronic health record. In Proceedings of the Conference on Artificial Intelligence in Medicine in Europe, Verona, Italy, 18–22 July 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–5.
19. Aramaki, E.; Miura, Y.; Tonoike, M.; Ohkuma, T.; Masuichi, H.; Waki, K.; Ohe, K. Extraction of adverse drug effects from clinical records. In Proceedings of the MEDINFO 2010: 13th World Congress on Medical Informatics, Cape Town, South Africa, 12–15 September 2010; IOS Press: Amsterdam, The Netherlands, 2010; pp. 739–743.
20. Rezaei, Z.; Ebrahimpour-Komleh, H.; Eslami, B.; Chavoshinejad, R.; Totonchi, M. Adverse drug reaction detection in social media by deep learning methods. *Cell J.* **2020**, *22*, 319–324.
21. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **2020**, *36*, 1234–1240. [CrossRef]
22. Mallick, C.; Das, A.K.; Dutta, M.; Das, A.K.; Sarkar, A. Graph-based text summarization using modified TextRank. In *Soft Computing in Data Analytics*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 137–146.
23. Partalidou, E.; Spyromitros-Xioufis, E.; Doropoulos, S.; Vologiannidis, S.; Diamantaras, K.I. Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy. In Proceedings of the 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Thessaloniki, Greece, 4–17 October 2019; pp. 337–341.
24. Gupta, V.; Bharti, P.; Nokhiz, P.; Karnick, H. SumPubMed: Summarization dataset of PubMed scientific articles. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop, Online, 5–6 August 2021; pp. 292–303.
25. Negi, K.; Pavuri, A.; Patel, L.; Jain, C. A novel method for drug-adverse event extraction using machine learning. *Inform. Med. Unlocked* **2019**, *17*, 100190. [CrossRef]
26. Park, S.; Kwak, N. Analysis on the dropout effect in convolutional neural networks. In Proceedings of the Asian Conference on Computer Vision, Taipei, Taiwan, 20–24 November 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 189–204.
27. Jacovi, A.; Shalom, O.S.; Goldberg, Y. Understanding convolutional neural networks for text classification. *arXiv* **2018**, arXiv:1809.08037.