Rumor Detection

Venkata Saideep Nagulapati
Lakehead University
Student ID: 1105142
vnagulap@lakeheadu.ca

Sai Rohit Rapelli
Lakehead University
Student ID: 1107345
srapelli@lakeheadu.ca

Rajesh Aytha
Lakehead University
Student ID: 1105343
raytha@lakeheadu.ca

Manoj Kumar Bhuma
Lakehead University
Student ID: 1101603
mbhuma@lakeheadu.ca

I. Abstract

This paper provides the background review, the exemplary models implemented and our proposed model on Rumor Detection. Rumors or fake news have become one of the important problems in our digital age. Now-a-days micro blogging websites or some of the news websites where the events and news being published are leading to the spread of rumors and fake news. So, verifying and validating this information is becoming challenging task. These rumors are mainly spreading through news websites and editorial columns. It is very difficult to detect these rumors as many different news columns are posted in very different websites through unverified sources every day. This paper gives an overview of a dataset of fake news's collected from different sources and machine learning techniques used segregate fake news and real news from that dataset. We also proposed a deep learning model which can detect the fake news from the dataset.

II. Introduction

A rumor is a statement or a column in a website, whose validity is questionable and has no clear source, even though its conceptual or partisan sources and motives are apparent. A rumor spreads rapidly over social media platforms since it is a place where huge amount of data is continuously generated. The field of rumor detection has gained interest in both academics and industries. It seems that government agencies are also putting their efforts to reduce the spread of fake news and rumors. Now-a-days, micro blogs are becoming more popular because the news, events first appear on these websites. The data from these sources are unverified and doesn't always gives the right information.

III. Possible Challenges

- **Comprehension Semantics:** Most rumors mislead the audience by providing false information in a disguised and mixed format. It is difficult for computer to understand these kinds of semantics.

- **Enormous Variations:** Rumors can spread across different types of languages and different topics. So, the algorithm needs to be trained in order to distinguish these kinds of rumors because, if we do limit data labelling, algorithm may fail.
- **Heterogeneous Structure:** People discuss and spread the rumors in social media. So, we need to understand the behaviour of these people, we need to construct the algorithm so that, it helps in rumor detection.

IV. Literature Review:

Vahed and Emily[3] developed a general framework which predicts whether a given tweet is a rumor and if so, whether the user believes it to be a rumor or not. The tweets are processed as they appear in users wall, there is no pre-processing. The approach is building different Bayes classifiers and learning a linear function of these classifiers for retrieval in first class and classification in the second.

Pavithra and Shibily[2] has focused their the work mainly on detecting rumours using neural network models such as CNN and RNN. This model mainly depends on text and structure.They proposed a system consisting of four modules. Which are;

- Corpus processing module
- Polarity Identification module
- Tree Structuring module
- Neural network model creation

The tree-based structuring is implemented by identifying child and parent nodes based on the tweet ID's.

V. Dataset

The dataset used contains three columns and 7795 rows. This dataset contains news editorials from different sources. The three columns in the dataset are described below.

- **Title:** The title or heading of the news published in a particular website.
- **Text:** The actual news corresponding to the tile in the form of text. The full editorial is present in the "text" column. The idea of the project is to identify whether this text or news is REAL or FAKE.
- **Label:** This field contains the information about whether the given news is REAL or FAKE.

VI. Existing Model

- **Vectorization as pre-processing :**
  Why vectorization?
  The corpus present in the dataset is in the form of text. But any models like linear regression, logistic regression, or neural network take numerical inputs. To convert text data into numerical forms, we use vectorization techniques. The vectorization techniques

used in the current existing model are:

- **Count Vectorization:** This method deals with counting the number of occurrences of each word in a document. CountVectorizer in sklearn library is used both for tokenization of words in the document and build a vocabulary of known words. After applying CountVectorizer, an encoded vector is returned with a length equal to the entire vocabulary and count for number of times the word appears in the document. As this returned vector contains many zeros, it is known as sparse representation. Python handles sparse vectors with scipy.sparse package.Its respective code is shown in Appendix A.

- **TF-IDF Vectorization:** TF-IDF stands for Term Frequency – Inverse Document Frequency.
  - ∗ The term frequency refers to how much a term (word) appears in a document.
  - ∗ Inverse document frequency refers to how common or rare a term appears in a document. Its respective code is shown in Appendix B.

- **Hashing:** Counts and frequencies can be very useful, but drawbacks of one of these approaches is that the vocabulary can become very growing. This, in effect, will allow massive vectors for document encoding and will place huge memory requirements and slow down algorithms. A smart job is to use a hash of words in one way to turn them into integer. The tricky part is that no vocabulary is required, and you can choose a fixed length vector that is arbitrary long. One downside is that hash is a one-way feature and there's no way to translate the encoding to a word. This approach is implemented by the HashingVectorizer class which can be used to tokenize and encode documents as needed. Its respective code is shown in Appendix C.

• **Models Used:** The model used in the existing model is multinomial naïve bayes.

  - **Naive Bayes :** Naive bayes algorithm is a family of probabilistic algorithms based on applying Bayes theorem with the "naive" assumption of conditional independence between every pair of features.

  - **Multinomial Naïve Bayes:** In this model, the sample represents the frequencies with which certain events are generated with multinomial. Its respective code is shown in Appendix D.

VII. Proposed Model

Our model follows the same preprocessing steps as in existing model. The difference is that, we have used convolution neural network for classification of the data.

Convolution Neural Network: Convolution Neural Network is a type of deep learning network which takes the input data, assigns weights and bias to various features in data so that we can differentiate between the data and guess the unseen data. A Convolution network mainly learns through filters.

- **Filters in Convolution:** A filter is a vector of weights which we multiply/convolve with input. The main function of a filter is to extract important features.

- **Pooling Layer:** Pooling layer is responsible for reducing spatial size of convolved feature. This is mainly done to reduce the computational complexity required to process the data through dimensionality reduction. There are two types of pooling, Average Pooling and Max Pooling.

- **Activation Functions:** An activation function is a node which is placed at the end or in between the layers. The main purpose of this is to decide when to fire the neuron. It helps in maintaining the output between a range of values. There are many activation functions. Some of them which are widely used are:
  - Sigmoid
  - ReLu
  - Sine

Its respective code is shown in Appendix E.

VIII. Environment Setting and Experimental Results

Test and Train Set: Test and trainset are divided in the ratio of 30 percent and 70 percent using *train_test_split* function from sklearn library.

Number of Layers: The model developed has 3 convolution layers, 3 max pooling layers, 1 input layer with added linear layers. The architecture is shown in Fig 1.

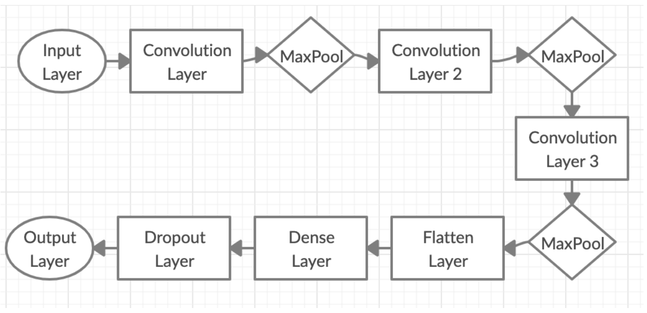

Fig. 1. Architecture of Neural Network

	Count Vectorization	TF-IDF	Hashing
Existing Model	0.893	0.857	0.868
Proposed Model	0.935	0.911	0.908

TABLE I
ACCURACY VALUES

Kernel Size: The kernel sizes are 3, 5 and 7 w.r.t each convolution layer.

Number of epochs: 25

Batch size: 64

Optimizer Used: Adam

Activation Function: ReLu

The experiment is done with different training parameters. The achieved results are tabulated in TABLE 1.

The comparison of the results is pictorially depicted in the Fig 2.

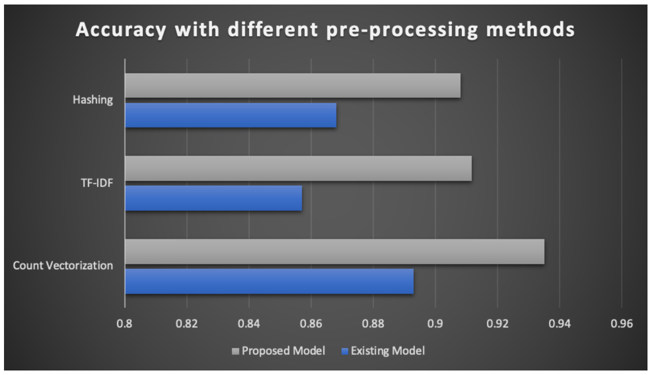

Fig. 2. Accuracy Comparison

IX. CONCLUSION

From the above experiment, the fake news dataset used in our model can be used to segregate fake and real news by using convolution neural network and various machine learning techniques there by successfully helping in detecting rumours. On comparison, we can also conclude that the usage of Convolution neural networks in classifying or detecting the rumors gives us best results than using simple machine learning algorithms.

Applications of Rumor Detection

- Helps in detecting rumours in social media.
- Helps in detecting deadly viruses in healthcare industry.
- To stop spreading of fake news in news paper industry

Github link:
https://github.com/saideep3/Test/blob/master/Project.ipynb

X. APPENDIX

A. Count Vectorizer

```
vectorozier = CountVectorizer(stop_words='english')
vectorizer_train = vectorozier.fit_transform(
    X_train.values)
vectorizer_test = vectorozier.transform(X_test.
    values)
```

Listing 1. Code for Count Vectorizer

B. TF-IDF

```
tfidf = TfidfVectorizer(stop_words='english',
    max_df=0.75)
tfVect_train = tfidf.fit_transform(X_train)
tfVect_test = tfidf.transform(X_test)
```

Listing 2. Code for TF-IDF

C. Hashing

```
from sklearn.feature_extraction.text import
    HashingVectorizer

hashing = HashingVectorizer(stop_words='english',
    n_features=5000)
hashing_train = hashing.fit_transform(X_train)
hashing_test = hashing.transform(X_test)
```

Listing 3. Code for Hashing

D. Existing model

```
clf = MultinomialNB()
clf.fit(hashing_train, y_train)
pred = clf.predict(hashing_test)
```

Listing 4. Model implemented in Existing system

E. Convolution Model

```
from tensorflow.python.keras.layers import Dense,
    Conv1D, Flatten, MaxPooling1D, Dropout
from tensorflow.python.keras import Sequential

#Model Initialization
model = Sequential()

def model_build(data_x):
  #Adding 1st convolution layer
  #with 32 filters and kernel_size = 3
  model.add(Conv1D(filters = 64, kernel_size=3,
    activation='relu', input_shape=(data_x.shape
    [1],1)))
  #Adding MaxPool layer with pool_size as 2
  model.add(MaxPooling1D(pool_size =2))
  #Adding 2nd convolution layer
  #with 64 filters and kernel_size = 5
  model.add(Conv1D(filters = 128, kernel_size=5,
    activation='relu'))
  #Adding MaxPool layer with pool_size as 2
  model.add(MaxPooling1D(pool_size =2))
  #Adding 3rd convolution layer
  #with 128 filters and kernel_size = 7
  model.add(Conv1D(filters = 256, kernel_size=7,
    activation='relu'))
  #Adding MaxPool layer with pool_size as 2
  model.add(MaxPooling1D(pool_size =2))
  model.add(Flatten())
  #Adding Fully connected layer
  model.add(Dense(128, activation='relu'))
  #Adding Dropout layer
  model.add(Dropout(0.5))
  #Output Layer with 5 outputs
  model.add(Dense(1, activation='sigmoid'))
```

Listing 5. Code for Convolution Layers

XI. CONTRIBUTIONS

The key contibutions of the team mates are listed below.

Sai Rohit Rapelli - Pre-Processing of data.

Rajesh Aytha – Implementing existing model.

Venkata Saideep Nagulapati – Creating proposed model.

Manoj Bhuma- Comparing results and reporting.

References

[1] Sardar Hamidian and Mona Diab, ”Rumor Detection and Classification for Twitter Data”, ArXiv, 2019

[2] Pavithra C P, Shibily Joseph, ”Deep Learning Approach For Rumour Detection In Twitter : A Comparative Analysis”, ICSEE, 2019

[3] Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, Qiaozhu Mei, ”Rumor has it: Identifying Misinformation in Microblogs”, ACL, 2011.

[4] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, “Rumor has it: identifying misinformation in microblogs,” In Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011, pp. 1589-1599.

[5] J. Harsin, “The Rumour Bomb: Theorising the Convergence of New and Old Trends in Mediated US Politics,” Southern Review: Communication, Politics and Culture. Issue 1, 2006, pp. 84-110.

[6] C. Ghaoui, “Encyclopedia of Human-Computer Interaction,” Encyclopedia of human computer interaction. IGI Global, 2005.

[7] A. Zubiaga, M. Liakata, R. Procter, G. W. S. Hoi, and P. Tolmie,Analysing how people orient to and spread rumours in social mediaby looking at conversational threads,PloS one, vol. 11, no. 3, p.e0150989, 2016.

[8] Y.-C. Chen, Z.-Y. Liu, and H.-Y. Kao, Ikm at semeval- 2017 task8: Convolutional neural networks for stance detection and rumorverification, inProceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 465469.

[9] C. Song, C. Tu, C. Yang, Z. Liu, and M. Sun, Ced: Credible earlydetection of social media rumors,arXiv preprint arXiv:1811.04175,2018.

[10] Gordon Allport and Leo Postman. 1947. The psychology of rumor. Holt, Rinehart, and Winston, New York.

[11] Galen Andrew and Jianfeng Gao. 2007. Scalable train- ing of l1-regularized log-linear models. In ICML ’07, pages 33–40.