EE-529 Project Report

Word2Vec+LSTM for Sentence Classification

Ahmed Ali Abbasi, Balaji S Pokuri

1 Problem Statement and Dataset Description

1.1 Objectives

The main goal of this project is to inverstigate the classification performance of a word2vec [1] + LSTM [2, 3] deep neural net architecture on the Food-101 dataset [4] and the Fakeedit dataset [5].

1.2 Data description

The Food-101 dataset has 101 classes, with each class corresponding to a food label. See Figure 1 for an example of data points and their corresponding class labels. The Fakkedit dataset [5] has binary, ternary and 6-way labels for each data point, with nearly 10^6 data points. Refer to section 4 for more details.

1.3 Relevant Works

Before outlining the challenges, we briefly list important relevant works:

- Bert-CNN ([6]): The deep learning architecture implemented in this repository is similar to the architecture we propose in this work, but we note the following important differences. i) The work [6] uses BERT embeddings for words while we plan to use word2vec. ii) The models are not trained on the the novel Fakeddit dataset we use. iii) The implementation is in TensorFlow, not in PyTorch, which is what we intend to use. Lastly, as we point out in the results section, switching from BERT to Word2vec greatly reduces the train and test time for the model. We explain the reason for this speed up in subsection 2.1.2.
- DistilFND ([7]): This repository uses the Fakeddit dataset and proposes a transformer based architecture, also with BERT embedding, for multi-modal learning in TensorFlow/Keras. As discussed earlier, we do not use the BERT embedding.
- Word2vec + LSTM based architectues [8, 9] have been considered for text classification problems on other datasets. We note that, to the best of our knowledge, we are the first to apply w2v+LSTM based architecture on the Food-101 and Fakeedit datasets.

There were two major challenges in this project. First was to gain familiarity with natural language processing techniques like tokenization, data cleaning (i.e. lower casing, removing hyphens and special characters etc). The second major challenge was to implment word2vec as as neural net layer. Doing so is necessary because it allows us to use existing libraries like Adam optimizer for optimizing the cross entropy loss.

text	food
Apple pie - Wikipedia	apple_pie
Apple Pie Oatmeal	apple_pie
Raspberry Apple Pie Pops | Driscoll's	apple_pie
Sour Cream Apple Pie Recipes | Recipebridge Re...	apple_pie
Crock-Pot Ladies Crock-Pot Apple Pie Moonshine	apple_pie
...	...
Pumpkin Waffles Recipe	waffles
Homemade Waffles	waffles
~wingerz » Overnight Waffles	waffles
Breads and Doughs-Pancakes and Waffles Recipes...	waffles
Rainbow Waffles recipe - from Tablespoon!	waffles

Figure 1: Examples of data points from two classes 'apple pie' and 'waffles'. Food-101 dataset has 101 classes. [4]

```
w2nModel(
  (word_embeddings): Embedding(22324, 100)
  (LSTM): LSTM(100, 64, batch_first=True)
  (drop1): Dropout(p=0.5, inplace=False)
  (FC1): Linear(in_features=64, out_features=256, bias=True)
  (drop2): Dropout(p=0.5, inplace=False)
  (FC2): Linear(in_features=256, out_features=101, bias=True)
  (Relu): ReLU()
)
```

Figure 2: Model summary.

1.4 Conclusions

The major findings of this work are

- Word2vec text embeddings achieve comparable performance to BERT embeddings, but greatly reduce the run-time of the model. For an explanation why, please see section 2.1.2.
- Using LSTMs, or recurrent neural networks, which model the temporal relataionship of words appearing in a sequence, is key to achieving good performance. Without the LSTM based embeddings, that is, by averaging the word2vec embeddings for all words in a sentence and training an SVM on the averaged embeddings, greatly reduces the resulting classification accuracy, which drops by 10 percent from 85 percent with LSTM to 75 percent without LSTM. See Table 2 for results.
- Preprocessing of text data to enforce the same length for all input sequences is an important hyperparameter/design consideration. The results included in this report show that too short a sentence length, that is too much truncation, and too long a sequence length, that is excessive zero padding, both degrade the overall classification performance of the network. See Table 1 for results.

2 Word2Vec + LSTM Based Neural Net for Classification

The summary of the proposed model is shown in Fig. 2. The Input is a sequence of integers

Modules	Parameters
LSTM.weight_ih_l0	76800
LSTM.weight_hh_l0	16384
LSTM.bias_ih_l0	256
LSTM.bias_hh_l0	256
bn.weight	64
bn.bias	64
FC1.weight	16384
FC1.bias	256
FC2.weight	1536
FC2.bias	6

Total Trainable Params: 112006

Figure 3: Total number of trainable paramters for the proposed model.

representing the words in a sentence. These integers can be thought of as the row indices of the embedding matrix output by training word2vec. The embedding matrix has rows equal to the number of words in the vocabulary and number of columns equal to to the dimension of the embedding. Word2vec layer outputs 100-dimensional embedding for each word. LSTM embeds the sequence into 64-dimensional embedding. It is relevant to note here that the output of the LSTM network is the output state of the final cell. We do not retain the hidden states or the output states of the other cells. We also use a dropout layer to regularize the network, that is to prevent over-fitting. A fully connected layer with 101 outputs is then fed into the cross-entropy loss. The total number of trainable parameters for this network is reported in 3. In the following sections, we briefly discuss the two main components of the network: word2vec and LSTM.

2.1 Word2Vec

We use the CBOW word2vec embedding model, which uses context, or neighboring words in the sentecnce, of a word to generate its embeddings. The inputs to the word2vec model are one-hot vectors. These vectors have entries corresponding to the indices of the neighbors equal to 1, rest zero. See example in 4. Overall, The input to word2vec is text data the output is a $V \times N$ vocabulary (or matrix). Each row of vocabulary is N-dimensional dense word embedding. Formally, The objective function maximized is

$$max_{W,W'} \Pr[w_t \mid w_c], \quad \Pr[w_t \mid w_c] = \frac{\exp(W'_j \frac{1}{c} \sum_{t=1}^{t=c} W x_t)}{\exp(W' \frac{1}{c} \sum_{t=1}^{t=c} W x)}. \tag{1}$$

Negative Sampling. We remark here that computing the denominator in (1), specifically multiplying by W', is numerically infeasible because $W' \in \mathbb{R}^{N \times V}$, where N is the total number of words in the text, which can be very large. To make the computation tractable, as subset of the rows of W' is multiplied. This technique is called negative sampling and details can be found in the original work [1].

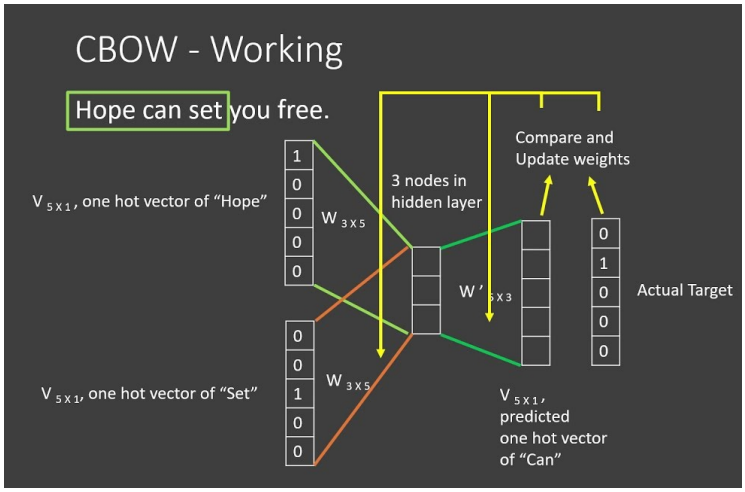


Figure 4: w2v is an unsupervised method for word embedding that learns embeddings by predicting the target word 'can' from from context words 'hope' and 'set'.

2.1.1 Word2Vec - trained on Food-101 Dataset

```
print(w2vModel.wv.most_similar('barbecue',topn=3))
print(w2vModel.wv.most_similar('apple',topn=3))
print(w2vModel.wv.most_similar('falafel',topn=3))
```

✓ 2.2s

```
[('babyback', 0.8852181434631348), ('bbq', 0.8723334074020386), ('breaded', 0.8309359550476074)]
[('pumpkin', 0.7679934501647949), ('shepherd', 0.7160881757736206), ('apples', 0.714442789554596)]
[('hummus', 0.8074979782104492), ('dressing', 0.8041724562644958), ('quinoa', 0.7597757577896118)]
```

Figure 5: The three most similar words to 'barbecue', 'apple', and 'falafel'.

We trained the word2vec algorithm on the entire dataset. The training took less than a minute. A few example embeddings are shown in Figure 5. The numerical values are the cosine distance between the word-embeddings.

2.1.2 Word2vec versus BERT

In this section, we will describe why switching from BERT to word2vec for text embeddings greatly reduces the run-time.

For the BERT layer, each input is a sentence. Numerically, each input (that is, each sentence) is mapped to three vectors: input_ids, mask_ids, segment_ids. Each of these vectors has length 40. Segment id has all entries zero, mask_id has non-padded entries 1, rest 0. Input-ids are integers. BERT embeds sentences into 40x768, where 40 is the user-defined max length of any sentence. Compare this to the word2vec model, which has just one input vector; this input vector is of dimension 32 and contains only integer numbers, compared to the BERT layer which has three input vectors of length 40 each. More importantly, word2vec embeds each sentence into a 32x100 dimensional-embedding, compared to a 40x768 dimensional embedding for BERT.

2.2 Long Short Term Memory (LSTM)

LSTMs [2, 3] are neural nets used when the input data are temporal. For example, ordered samples from an analog to digital converter, frames in a video sequence, or words in a sentence. This is

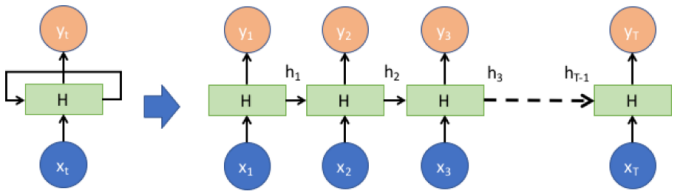


Figure 6: The mapping from x_1 to y_1 is the standard feed-forward neural network. The black arrows pointing right indicate that y_t, the output of cell t, depends on prior inputs $x_1, \cdots, x_{t-1}$.

because the final output of the LSTM networks depends on all previous inputs, in contrast to the Feedforward networks, which would treat each element in a sequence as a different input. Figure 6 illustrates this concept further.

3 Results for Food-101 Dataset

3.1 Preprocessing

The input data consitis of sentences with varying lengths, see figure 7. We preprocess the input data as follows: Truncate all sequences with length > threshold; zero-pad otherwise. Results in table 1 show that optimal sequence length is 16. This suggests that excessive padding or truncation both degrade performance. We also preprorcess the text sentences to remove punctuation, special characters and convert to lower-case.

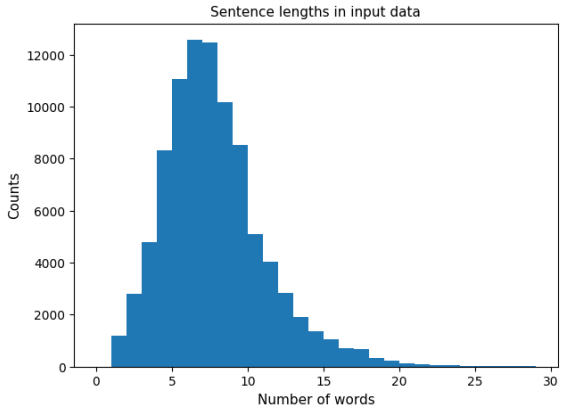


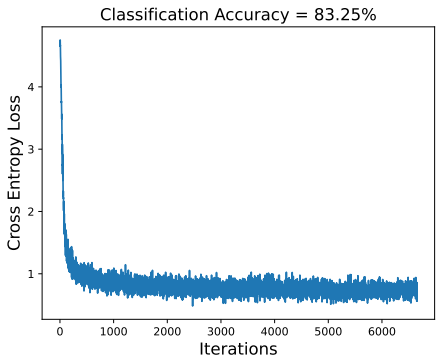
Figure 7: Input sentences have varying lengths. Most sentences are of length 16 or shorter.

3.2 Results

Table 1 shows the effect of changing the length of the input sequences and the dimension of the word2vec embedding on the classification accruracy. For a discussion of the resutls, please see section 1.4. The corresponding loss plot shows the cross entropy loss decreasing as a function of the iterations. Note that we used tha Adam optimizer which uses stochastic gradient descent with momentum to miniize the cross-entropy loss function. In table 2, we compare the proposed architecture with several benchmark methods. For a discussion of the results, please see 1.4.

Dimension	Length	Accuracy %	Runtime (seconds)
50	16	83	$\simeq 45$
50	32	82	$\simeq 45$
100	16	82	$\simeq 45$
100	32	79	$\simeq 45$

Table 1: Dimension is the dimension of the word2vec embedding. Length is the input sequence length.



Loss plot using Adam optimizer for stochastic gradient descent.

Method	Accuracy %	Runtime
w2v-lstm-softmax	83	45 seconds
w2v-lstm-svm	$\simeq 82$	10 minutes
w2v-lstm-kernel svm	$\simeq 82$	20 minutes
w2v-(avg)-linear svm	$\simeq 75$	10 minutes
bert-lstm-softmax	84	$\simeq 10$ hours
vgg-net	78	5 minutes

Table 2: bert-lstm results reported in [4]. vgg-net results reported in [10]. Note that i) the results deteriorate significantly if LSTM is removed from the model, see section 1.4 for discussion, and ii) word2vec based model provides comparable performance to the BERT-based model with a significantly smaller run-time (see section 2.1.2 for discussion).

4 Problem Statement 2

Fakeddit is a multi-modal dataset consisting of over 1 million samples from multiple categories of fake news collected from social media platform Reddit. After being processed through several stages of review, the samples are labeled according to 2-way, 3-way, and 6-way classification categories through distant supervision. Each data entry consists of a post title, comments made by other users under the post, and optionally an image depicting the post. While the dataset is multi-modal, in this project we use only the post titles to classify into two or six classes.

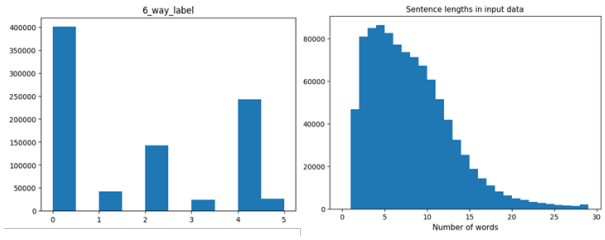


Figure 8: Distributions of word length in Fakeddit

After filtering out unlabelled data and posts with empty titles, the dataset has 887,325 training data and 84,481 testing data points. The distribution of word-lengths of posts is shown in figure 8, and sample data points for each of the six classes (1 Real and 5 Fake classes) are shown in figure 9.

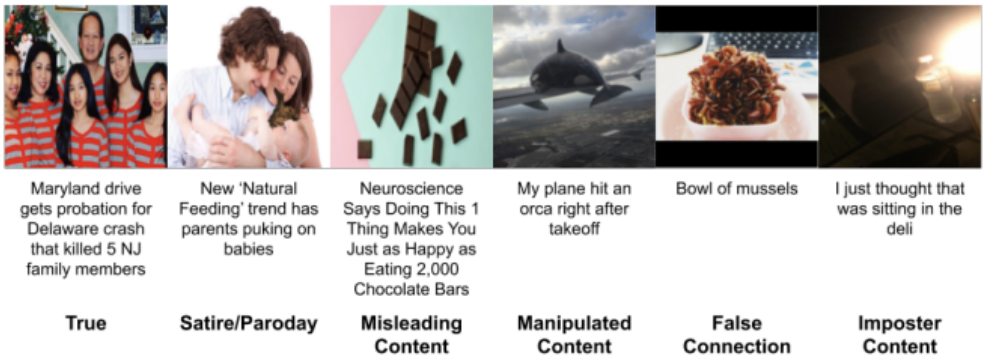


Figure 9: Dataset examples with 6-way classification labels

5 Model Architecture

The original paper uses BERT and Inferset models for classifying the data into Real or different fake classes. However, as we have shown with Food-101 dataset, the model complexity and resource demands of training and inferring from the model is too high. In order to compare with simpler models for text processing, we consider TF-IDF embedding followed by softmax as simplest text classifying model, and compare it with increasingly complex models. Finally, we train the Word2Vec + LSTM + SoftMax model and compare it with published results from BERT and Inferset. A high-level view of different architectures we tested are shown in figure 10. Note that TF-IDF + Kernel Logistic Regression did not show any improvement over TF-IDF with Softmax (i.e., multinomial Logistic Regression) , and TF-IDF with Kernel-SVM took too long to converge. Subsampling the training dataset randomly to get equal number of samples for classes led to loss in accuracy, it has not been reported.

Model implementation details are as follows:

- TF-IDF (sklearn): output embedding pruned to top 300 words

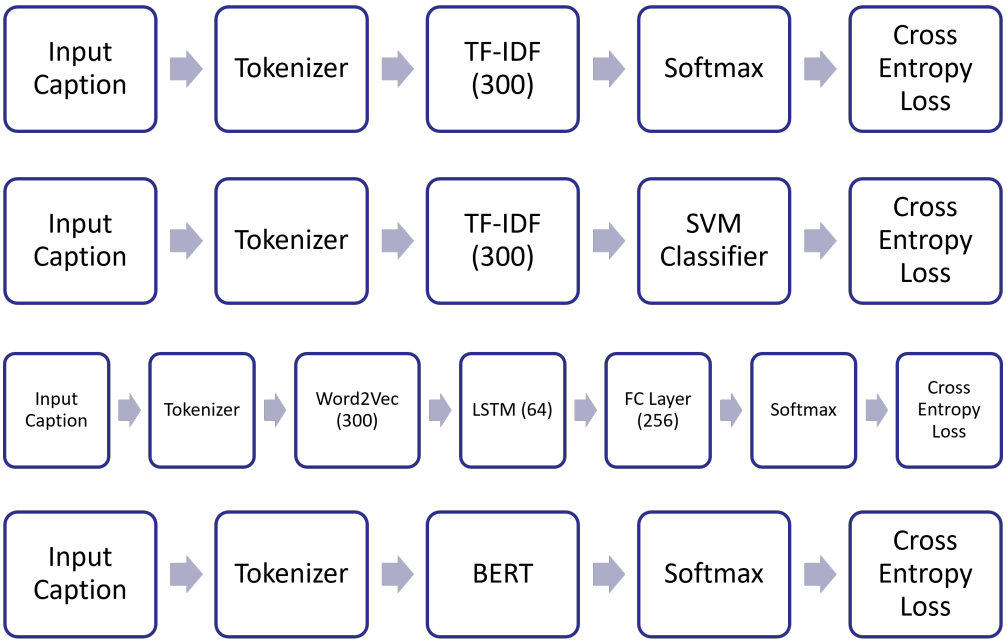


Figure 10: Model Architectures

- Logistic Regression (sklearn): 5-fold cross validation
- Word2Vec (pytorch): 300-dimension vector outputs
- LSTM (pytorch): 64 cells, the word embedding prediction of last LSTM cell ($d = 300$) is used as output
- Loss function: Cross-entropy loss

6 Results for Fakeedit Dataset

The Word2Vec+LSTM model shows a very fast training phase, as can be seen in figure 11 that objective function decays within first few iterations of the first epoch. Although we continue to train the model for 10 epochs, the decrement in cross entropy loss is marginal after the first few hundred iterations.

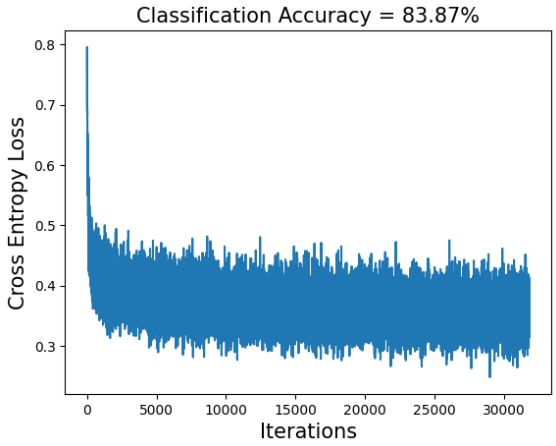


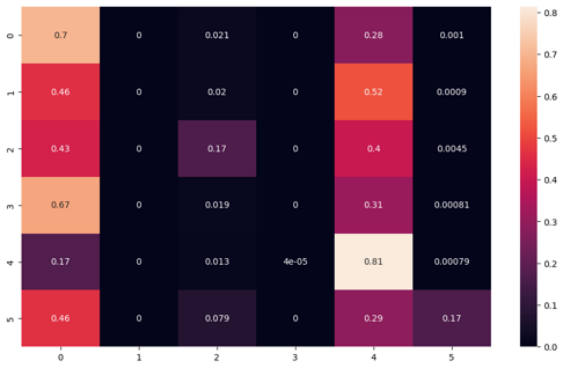
Figure 11: Training Loss for 10 epochs * 1592 batches per epoch

The model accuracy on testing dataset of different models have been provided in table 3. In addition to accuracy on the test dataset, we also report the confusion matrix on testing dataset in figure 12. We see that in case of binary classification, off-diagonal elements have almost equal rates, but in 6-way classification, most of the predictions are either as Real or False connection, the most common classes in the training dataset. Training the model on data with equal frequencies of all classes makes the confusion matrix to have increased mass on diagonal entries, but the overall accuracy of the model reduces drastically, as can be seen in figure 12d.

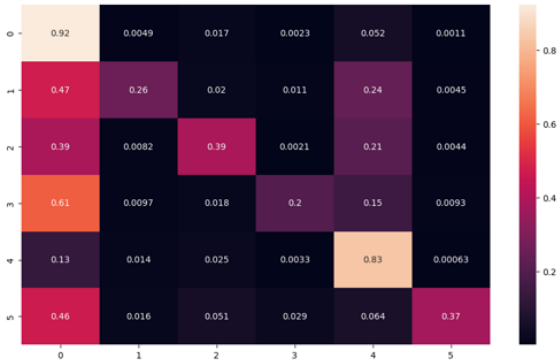
Method	2-way Accuracy(%)	6-way Accuracy(%)
TFIDF-Softmax	70.74	57.66
TFIDF-SVM	70.70	57.53
Word2Vec-LSTM-NN-Softmax	83.66	73.45
Word2Vec-LSTM-NN-Softmax with balanced classes	83.87	56.69
BERT-Softmax	86.44	76.77
Inferset-Softmax	86.31	76.66

Table 3: BERT and Inferset results reported in original paper ([11])

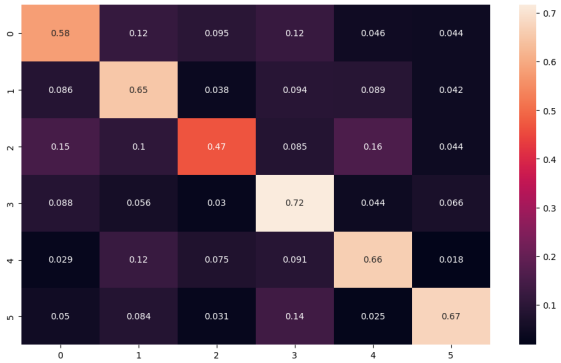
(a) TF-IDF with Logistic Regression



(b) TF-IDF with Linear SVM



(c) Word2Vec LSTM NN



(d) Word2Vec LSTM NN with balanced classes

Figure 12: Confusion Matrix of different models for 6-class classification

References

[1] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[2] R. C. Staudemeyer and E. R. Morris, "Understanding lstm – a tutorial into long short-term memory recurrent neural networks," 2019.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] I. Gallo, G. Ria, N. Landro, and R. L. Grassa, "Image and text fusion for upmc food-101 using bert and cnns," in *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2020, pp. 1–6.

[5] S. L. Kai Nakamura and W. Y. Wang, "Github repo for fakeedit dataset." [Online]. Available: https://github.com/entitize/Fakeddit

[6] "Image-and-text-fusion-for-upmc-food-101-using-bert-and-cnns. url: https://github.com/artelab/image-and-text-fusion-for-upmc-food-101-using-bert-and-cnns."

[7] "Explanatory detection of fake news with deep learning. url: https://github.com/enzomuschik/distilfnd."

[8] Y. Sung, S. Jang, Y.-S. Jeong, J. Hyuk *et al.*, "Malware classification algorithm using advanced word2vec-based bi-lstm for ground control stations," *Computer Communications*, vol. 153, pp. 342–348, 2020.

[9] P. F. Muhammad, R. Kusumaningrum, and A. Wibowo, "Sentiment analysis using word2vec and long short-term memory (lstm) for indonesian hotel reviews," *Procedia Computer Science*, vol. 179, pp. 728–735, 2021.

[10] "Vgg16 re-trained model: 78% accuracy. url: https://www.kaggle.com/code/josephmiguel/vgg16-re-trained-model-78-accuracy."

[11] K. Nakamura, S. Levy, and W. Y. Wang, "r/fakeddit: A new multimodal benchmark dataset for fine-grained fake news detection," 2020.