**Report on Fake News Detection Using Embedding's and RNN Architectures**

**Natural Language Processing**



**Session 2024-2026**

**By**

Jamal Shah

**Submitted To**

Dr. Atif Khan

Master of Science in Data Science

2nd Semester (Group A)

**Report on Fake News Detection Using Embedding's and RNN Architectures**

**1. Dataset Description**

The dataset used to perform this task was "Fake News Detection Dataset". A set of both real and forged news articles of the real-world source. Split into two data files:

**True.csv:** Included only the authenticated news articles.

**Fake.csv:** Included misleading or forged information articles.

**The Characteristics of datasets:**

**True.csv:**  21,417 lines labeled as "1" real news.

**Fake.csv:**   23,481 lines labeled as "0". Fake news.

Both the datasets have columns named title, text, subject, and publication date. These were merged together and preprocessed into a unified dataset for the analysis.

**2. Data Preprocessing Techniques**

These preprocessing techniques are applied to text data:

**1. Text Cleaning:**

* Removing punctuation, special characters, and unwanted symbols.
* Lowercase text.
* Text cleaning using the NLTK Stopwords list.

**2. Tokenization:**

* The text splits into sequences for embedding layers

**3. Padding:**

* All sequence padded to maintain uniform length.

**4. Data Splitting**

* Apply train\_test\_split to split 70% in training, and 15 % each in validation, and test from the dataset.

**3. Architecture of the models**

**Embeddings used:**

**1. Pre-Trained Embeddings**

* **Word2Vec:** Took a pre-trained model from the Google News dataset and used its 300 dimension version.
* **FastText:** Implemented by using its pre-trained wordpiece embeddings coming from the Wiki News dataset
* **GloVe:** Loaded vectors 100d by Stanford NLP

**2. Custom- Trained Embeddings:**

* Training Word2Vec, FastText, GloVe on a custom dataset.

**RNN Architectures:**

Three types of the recurrent neural network are used.

**1. LSTM (Long Short-Term Memory):**

* Two LSTM layers with dropout.
* Output layer employed with sigmoid to be used for binary classification.

**2. GRU (Gated Recurrent Units):**

* Two GRU layers with roughly the same dropout setup as LSTM.

**3. Bi-LSTM (Bidirectional LSTM):**

* Bidirectional LSTM for contextual information in both directions.

Embedding layers pretrained were used as an input to the RNN models.

**4. Performance Metrics**

The models were evaluated using:

1. **Accuracy**: Proportion of correct predictions.
2. **Precision**: Correct positive predictions divided by all positive predictions.
3. **Recall**: Correct positive predictions divided by actual positives.
4. **F1-Score**: Harmonic mean of precision and recall.

**Pretrained Embeddings Results:**

|  |  |  |
| --- | --- | --- |
| Embedding | Loss | Accuracy |
| Word2Vec | 0.147 | 96.57% |
| FastText | 0.030 | 99.27% |
| GloVe | 0.032 | 99.48% |

**Custom-Trained Embeddings Results:**

|  |  |  |
| --- | --- | --- |
| Embedding | Loss | Accuracy |
| Word2Vec | 0.061 | 98.78% |
| FastText | 0.018 | 99.73% |
| GloVe | 0.265 | 92.23% |

**5. Observations and Conclusions**

1. **Pretrained vs. Custom-Trained Embeddings**:

* FastText consistently outperformed other embeddings in both pretrained and custom-trained settings due to its subword information capability.
* Custom-trained embeddings adapted better to the dataset compared to pretrained Word2Vec but were slightly behind pretrained GloVe.

1. **RNN Architectures**:

* **Bi-LSTM** achieved the best results among the RNNs due to its ability to process sequences bidirectionally, capturing richer contextual information.
* **GRU** performed comparably to LSTM but with slightly faster training times due to its simpler structure.
* **LSTM** excelled in handling longer dependencies but required more training time.

**6. Discussion Points**

**Embedding Performance:**

* **FastText** was the most effective embedding technique, as it captured morphological details and performed well across all metrics.
* **GloVe** pretrained embeddings were competitive, but the custom-trained version underperformed, possibly due to the dataset's size and vocabulary limitations.
* **Word2Vec** pretrained embeddings were robust but slightly less effective than FastText and GloVe.

**Impact of RNN Architectures:**

* The **Bi-LSTM** model's bidirectional processing provided enhanced accuracy and context awareness, especially for nuanced text sequences.
* **GRU** was a good trade-off between performance and computational efficiency.
* **LSTM** was effective but computationally intensive.

**7. Comparative Analysis**

**Traditional Embeddings (Word2Vec, FastText, GloVe):**

**Advantages**:

* Pretrained models save computational effort.
* Easy to integrate into various architectures.

**Limitations**:

* Vocabulary dependency: Words not present in the pretrained corpus are not represented effectively.
* Limited contextual understanding compared to transformer-based embeddings.

**Transformer-Based Embeddings (BERT):**

**Advantages**:

* Contextual embeddings consider the surrounding text.
* Handles out-of-vocabulary words using subword tokenization.

**Limitations**:

* Computationally expensive.
* Requires more fine-tuning compared to traditional embeddings.

Transformer-based embedding’s like BERT outperform traditional methods in nuanced text analysis due to their contextual understanding. However, for this dataset, pretrained FastText embeddings combined with Bi-LSTM achieved similar results at a lower computational cost.