Unraveling Misinformation: LSTM-based Classification of Political Statements

Aarush Kartik

Insprit AI

October 20, 2024

Outline

Introduction

- The rapid spread of misinformation during events such as presidential elections has intensified the need for reliable fake news detection methods.
- Misinformation about political candidates, their background, and affiliations tends to spread across social media and news outlets.
- Such misinformation complicates decision-making processes for individuals and groups, potentially undermining democratic processes.
- Machine learning techniques have emerged as powerful tools to tackle the challenge of identifying and mitigating fake news.

Literature Review

- Key works in your field
- Gaps in existing research
- Positioning of your work in relation to existing literature

Problem Statement

- Define the problem you are addressing
- Why this problem is important
- Key challenges involved

Methodology

- Approach taken to solve the problem
- Data collection and experimental setup
- Models, algorithms, or frameworks used
- Steps involved in your research

Dataset Description

- **LIAR Dataset** [?]
  - 12.8K manually labeled short statements
  - Sources include political debates, TV ads, social media posts
  - Each statement annotated with one of six truthfulness ratings:
    - `true`
    - `mostly-true`
    - `half-true`
    - `barely-true`
    - `false`
    - `pants-fire`
  - Includes metadata:
    - Subject
    - Speaker
    - Job Title
    - State
    - Party Affiliation

Data Acquisition

- Obtained from the official repository [**?**]
- Developed an automated script for:
  - Downloading and extracting the dataset
  - Ensuring reproducibility and efficiency
  - Checking for local existence to avoid redundant downloads
- Dataset format:
  - Tab-separated values (TSV)
  - Split into training, validation, and test sets

Data Preprocessing

- Performed several preprocessing steps:
  - Data loading and column renaming
  - Statement length computation
  - Label encoding
  - Textual data transformation
  - Data reshaping for LSTM input
  - One-hot encoding of labels

Data Loading and Column Renaming

- Used `pandas` library to read TSV files into DataFrames
- Renamed columns for clarity:
  - **ID**: Unique identifier
  - **Label**: Truthfulness rating
  - **Statement**: Text content
  - **Subject**, **Speaker**
  - **Job Title**, **State Info**
  - **Party Affiliation**

Statement Length Computation

- Calculated length of each statement
- Purpose:
  - Understand distribution of statement lengths
  - Assess complexity and variability of textual data
- Computation:

$$l_i = \text{len}(s_i) \tag{1}$$

- Where:
  - l_i: Length of the i-th statement
  - s_i: Text content of the i-th statement

Label Encoding

- Converted categorical labels to numerical values
- Mapping:

$$\begin{aligned}\texttt{true} &\to 0\\ \texttt{mostly-true} &\to 1\\ \texttt{half-true} &\to 2\\ \texttt{barely-true} &\to 3\\ \texttt{false} &\to 4\\ \texttt{pants-fire} &\to 5\end{aligned}$$

- Reflects ordinal relationship among labels
- Encoded label for i-th statement: y_i

Textual Data Transformation

- Employed Bag-of-Words (BoW) model using Count Vectorization
- Transformed each statement s_i into vector $\mathbf{x}_i \in \mathbb{R}^V$
- V: Size of vocabulary from training data
- Components:
$$x_{ij} = \text{Count}(\text{word}_j, s_i) \tag{2}$$
- Where x_{ij}: Frequency of j-th word in i-th statement

Data Reshaping for LSTM Input

- LSTM expects input in 3D format: (samples, timesteps, features)
- Reshaped feature vectors accordingly:
  - Number of timesteps set to 1
  - Each statement treated as a single timestep with V features

One-Hot Encoding of Labels

- Converted numerical labels y_i into one-hot encoded vectors $\mathbf{y}_i \in \mathbb{R}^6$
- Representation:

$$\mathbf{y}_i = [y_{i0}, y_{i1}, y_{i2}, y_{i3}, y_{i4}, y_{i5}] \tag{3}$$

- Where:

$$y_{ij} = \begin{cases} 1 & \text{if } j = y_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- Suitable for multi-class classification with categorical cross-entropy loss

Data Splitting

- Dataset already partitioned by authors
- Splits:
  - **Training Set**: Train model parameters
  - **Validation Set**: Hyperparameter tuning, prevent overfitting
  - **Test Set**: Assess final performance

Summary of Data Preparation

1. **Data Acquisition**: Automated download and extraction
2. **Data Loading**: Read TSV files into DataFrames
3. **Data Cleaning**: Renamed columns, ensured data integrity
4. **Feature Engineering**:
   - Computed statement lengths
   - Transformed text into numerical features using BoW
5. **Label Encoding**: Converted labels to numerical and one-hot formats
6. **Data Reshaping**: Adjusted data shape for LSTM input

- Prepared data optimally for training the LSTM network
- Facilitated efficient learning and improved potential for accurate classification

Model Selection and Description

- Employed a **Long Short-Term Memory (LSTM)** network for fake news detection.
- Dataset: *Liar, Liar Pants on Fire* [**?**].
- LSTM networks are a type of Recurrent Neural Network (RNN) capable of learning long-term dependencies.
- Suitable for processing natural language text due to ability to capture sequential patterns.

Model Architecture

- Designed to effectively capture sequential patterns in textual data.
- Consists of the following layers:
  1. **First LSTM Layer**
     - 50 units
     - `return_sequences=True`
     - Returns hidden state output for each input time step.
  2. **First Dropout Layer**
     - Dropout rate of 0.7
     - Prevents overfitting by randomly deactivating 70% of neurons during training.
  3. **Second LSTM Layer**
     - 50 units
     - Processes sequence output from previous layer.
  4. **Second Dropout Layer**
     - Dropout rate of 0.7
  5. **Dense Output Layer**
     - Fully connected layer with 6 units (corresponding to 6 classes).
     - Uses softmax activation function.

Model Compilation and Training

- Compiled with:
  - **Loss Function**: Categorical Cross-Entropy
  - **Optimizer**: Adam optimizer [?]
  - **Learning Rate**: 0.001
- Trained for up to 30 epochs.
- Implemented **Early Stopping** based on validation loss to prevent overfitting.

Mathematical Formulation (LSTM Equations)

LSTM Units are defined by the following equations at each time step t:

$$\text{Input Gate: } i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$

$$\text{Forget Gate: } f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6}$$

$$\text{Cell Candidate: } \tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{7}$$

$$\text{Cell State Update: } C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{8}$$

Mathematical Formulation (Continued)

$$\text{Output Gate: } o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (9)$$

$$\text{Hidden State: } h_t = o_t \odot \tanh(C_t) \quad (10)$$

Where:

- x_t: Input vector at time t
- h_{t-1}: Hidden state from previous time step
- i_t, f_t, o_t: Input, forget, and output gates
- C_t: Cell state at time t
- $\tilde{C}_t$: Candidate cell state
- σ: Sigmoid activation function
- tanh: Hyperbolic tangent function
- $\odot$: Element-wise multiplication
- W, U: Weight matrices
- b: Bias vectors

Hyperparameters and Regularization

- **Number of LSTM Units**: 50 units in each LSTM layer
- **Dropout Rate**: 0.7 after each LSTM layer
- **Optimizer**: Adam optimizer
- **Learning Rate**: 0.001
- **Loss Function**: Categorical Cross-Entropy
- **Early Stopping**:
  - Training halted if validation loss does not improve for 5 consecutive epochs
  - Helps prevent overfitting

Model Training and Evaluation

- **Training Procedure**:
  - Mini-batch gradient descent
  - Batch size: 32
  - Monitored performance on validation set
  - Restored best model weights using early stopping
- **Evaluation Metrics**:
  - **Accuracy**: Overall correctness
  - **Confusion Matrix**: Detailed breakdown per class
  - **Classification Report**:
    - Precision
    - Recall
    - F1-score

Rationale for Model Selection

- LSTMs are effective for modeling sequential data and capturing long-term dependencies.
- Crucial for understanding context and nuances in natural language.
- Dropout layers serve as regularization to prevent overfitting.
- Aimed to generalize well to unseen data by learning robust features.

Implementation Details

- Implemented using **TensorFlow Keras API**.
- **Model Definition**:
  - Used `Sequential` API to stack layers linearly.
- **Callbacks**:
  - Early stopping implemented using `EarlyStopping` callback.
  - Monitored validation loss.
- **Evaluation Functions**:
  - Custom methods defined for evaluation.
  - `evaluate` method computes accuracy, confusion matrix, classification report.

Results

- Evaluated the LSTM model against several baseline models.
- Utilized standard metrics:
  - Accuracy
  - Precision
  - Recall
  - F1-score
- Marginal improvements in accuracy observed.
- Precision and recall values were significantly lower than anticipated.

LSTM Model Performance

- **Precision**: 0.21
- **Recall**: 0.20
- Significant challenges in identifying true positive instances of fake news.
- Model struggled to minimize false negatives.
- Low precision suggests legitimate news articles were misclassified as fake.

Confusion Matrix

	true	mostly-true	half-true	barely-true	false	pants-fire
true	62	37	41	17	41	10
mostly-true	67	46	48	34	38	8
half-true	48	53	60	47	39	18
barely-true	30	28	46	46	45	17
false	45	32	46	36	74	16
pants-fire	13	9	20	15	17	18

Table: Confusion Matrix: True Labels vs. Predicted Labels

Analysis of Results

- Performance underscores the complexity of fake news detection.
- Highlights the need for more nuanced feature extraction.
- Suggests re-evaluation of the training data.
- Indicates that relying solely on textual features may not suffice.

Conclusion

- Explored the effectiveness of an LSTM model for fake news detection.
- Model showed marginal improvement in accuracy over traditional methods.
- Overall performance was disappointing, particularly in precision and recall.
- Results reveal complexities inherent in this challenging task.

Future Work

- Need for a more robust approach to fake news detection.
- Consider incorporating various data sources and model architectures.
- Experiment with hybrid models combining LSTMs with:
  - Convolutional Neural Networks (CNNs)
  - Graph-based models
- Aim to capture both sequential and relational features inherent in news articles and their dissemination networks.

Acknowledgments

- Thanks to your advisors, collaborators, and mentors
- Acknowledge any funding or institutional support

Questions

Thank you!

Questions?