Text Classification using Recurrent Neural Network

Tomeh Lara

The Faculty of Mathematics and Computer Science, University of Bucharest, Bucharest, Romania

Abstract: in this project, we develop an RNN models for the task of text classification on the AG news dataset (**Zhang et al, 2015)** [1] which consists of news articles divided into four categories: business, science and technology, sports, and world. We achieve a final accuracy of 0.905 on the test set. This result indicates that the model was able to effectively classify the news articles in the AG news dataset.

Analysis of main idea: the main idea of this project is to evaluate the effectiveness of Recurrent Neural Networks (RNN) for the task of text classification. The high accuracy achieved by the model on the AG news dataset suggests that LSTM networks are a promising approach for this type of problems.

Why RNN instead of CNN for text classification? CNNs are a type of neural network that are commonly used for image classification tasks. They are able to learn spatial hierarchies by repeatedly applying convolutional filters to the input data and pooling the resulting feature maps. However, CNNs are not as well-suited for text classification tasks. This is because the structure of text data is very different from that of images. Text is sequential and ordered, while images are grid-like and unordered. Therefore, the techniques that work well for image data, such as convolution and pooling, are not as effective for text data. On the other hand, RNNs are specifically designed to handle sequential data and process the context of a word in relation to the words before it, which is important in NLP tasks like text classification. They are able to capture long-term dependencies in the data by using

gating mechanisms to control the flow of information through the network. This makes RNNs a more natural choice for text classification tasks due to the unique properties of text data.

Related work:

1. Meng et al (2020) [2]. present the **LOTClass** model, a label-name-only text classification approach that is built by define a category vocabulary for each class. **LOTClass** has been evaluated on four benchmark text classification datasets (AG News, DBPedia, IMDB, Amazon corpora) and achieved 0.864 accuracy without learning from any labeled data. It only requires the use of at most 3 words per class as the label name, significantly outperforming existing weakly-supervised methods and even demonstrating comparable performance to strong semi-supervised and supervised models.
2. Nyandwi, J. [3] Design long short-term memory model to classify the ag news into four categories, and he tries a classical model and another one with more stacked LSTM layers in addition to Gate Recurrent Unit model (GRU). He gets an accuracy 0.897 in LSTM model, 0.903 in GRU model.

Models and Methodology:

We use both types of RNNs, they differ in their architectures and the way they store and process information. LSTMs use memory cells, gates that regulate the flow of information into and out of the cells, and hidden states that preserve information across time steps. This allows LSTMs to better capture and preserve long-term dependencies in sequential data. GRUs, on the other hand, use a simpler architecture that combines the input and forget gates into a single update gate and merges the memory cell and hidden state into a single hidden state. In terms of performance, both LSTMs and GRUs have been shown to perform well on a wide range of NLP

tasks, and the choice between the two often comes down to the specific requirements of the task and the availability of data. In summary, LSTMs and GRUs are both types of RNNs designed to handle long-term dependencies in sequential data, but they differ in their architectures and the way they store and process information.

Dataset: the dataset used is the AG news dataset (**Zhang et al, 2015)** [1] which is a large collection of news articles from various sources, totaling more than one million articles. The dataset is divided into four categories: World, Sports, Business, and Science and Technology. There are a total of 120,000 training samples and 7,600 testing samples, with each class containing 30,000 training samples and 1,900 testing samples. This dataset is well-suited for testing and evaluating the performance of machine learning models on text classification tasks.

Implementation: In this report, we will describe the process of using an LSTM and GRU to classify text data. We will begin by preprocessing the data. We will then create the model using the Keras API, train the model, and evaluate its performance. Finally, we will discuss the results and conclude with a summary of our findings.

In order to use this dataset for machine learning purposes, we first preprocess the data by converting all text to lowercase and removing punctuations, digits, non-word characters, extra spaces, contraction, and stop words, finally stemming the words using Porter stemmer. This is a common preprocessing step for natural language processing tasks, as it helps to reduce the size of the vocabulary and eliminate some of the noise in the data. We then split the preprocessed data into three sets: a training set, a validation set, and a test set. The training set is used to train the model, the validation set is used to tune the model's hyperparameters, and the test set is used to evaluate the final performance of the model. In order to prepare the data, we first use a tokenizer to convert the text into sequences of integers. A tokenizer is a tool that processes the text, breaking it up into individual tokens or words and assigning each

word a unique integer identifier. This process allows us to represent the text as a series of integers and ensured that the input data had a consistent shape, which can be input to the model. We then use the pad sequences function from Keras API to standardize the length of the sequences. This is necessary because the model expects input data to have a fixed shape. By padding the sequences with zeros, we are able to ensure that all of the sequences have the same length and could be input to the model.

The code defines an LSTM model consists of the following layers:

- Embedding layer: This layer converts the input data, which is a sequence of integers representing the words in the text, into dense vectors of fixed size.
- Bidirectional LSTM layers: These layers are responsible for processing the input data and capturing long-term dependencies in the sequences. The parameter (return sequences) which is set to true specifies that the output of these layers should be a sequence, rather than a single vector.
- Global max pooling layer: This layer reduces the dimensionality of the data by taking the maximum value over all the time steps in the input sequence.
- Fully connected (dense) layers: These layers apply a non-linear transformation to the output of the previous layer and generate the final predictions.
- Output layer: This layer has four units, corresponding to the four categories in the dataset, and a SoftMax activation function, which converts the outputs to probabilities.

The other model is GRU consists of the following layers:

- Embedding layer: This layer maps the input data (which is an integer-encoded representation of the text) into dense vectors of fixed size. The

mask zero argument is set to True, which means that padding is used to handle sequences of different lengths.

- GRU layers: It uses gating mechanisms to control the flow of information. The argument return sequences is set to True, which means that the GRU layer will return the full sequence of outputs for each time step. The dropout and recurrent dropout arguments are regularization techniques used to prevent overfitting.
- Output dense layer: This is the output layer with 4 units and a SoftMax activation function. The SoftMax function maps the outputs to a probability distribution over the 4 classes.

We use the Adam optimization algorithm and sparse categorical cross-entropy loss for training. The final accuracy of the LSTM model on the test set is 0.905%, and the GRU model is 0.87%. This is a strong result, indicating that the LSTM model is able to effectively classify the news articles in the Ag News dataset.

Method	**Accuracy on Test data**
LOTClass (Meng et al., 2020) [2]	0.864
LSTM (Nyandwi, J) [3]	0.8974
GRU (Nyandwi., J) [3]	0.903
GRU (Tomeh, L)	0.8732
LSTM (Tomeh, L)	**0.905**

Table 1 Accuracy score on test set for all mentioned approaches

	LSTM model		**GRU model**	
Dataset	**Loss**	**Accuracy**	**Loss**	**Accuracy**
Training Data	0.228	0.929	0.352	0.884
Validation Data	0.299	0.904	0.384	0.874
Test Data	0.305	**0.905**	0.381	0.8732

Table 2 Loss and Accuracy for three sets

Conclusions and future work: overall, the process of creating and training an RNN model for the Ag News dataset was successful. The LSTM model achieve a high level of accuracy and could potentially be used for further analysis or classification tasks in the related fields. These results suggest that LSTM networks are a promising approach for text classification tasks, while the GRU model requires massive computing power, takes longer time, and produces less quality results than the latter. It is useful to test different possible directions for further research or improvement on the project like exploring other algorithms for text classification, such as transformer models, and comparing their performance to that of the LSTM model, fine-tuning the hyperparameters of the LSTM model, such as the embedding size and the number of units in the fully connected layers, to improve the model's performance, applying the model to other text classification datasets to test its generalizability, or adding additional features, such as word embeddings or entity recognition, to the model to see if they improve performance.

References:

1. Zhang et al (2015). ComeToMyHead. *AG news dataset.*
2. Meng et al. (2020). *Text Classification Using Label Names Only: A Language Model Self-Training Approach*. USA.
3. Nyandwi, J. *Complete Machine Learning Package*. https://nyandwi.com/machine_learning_complete/33_recurrent_neural_networks/.