ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
FACULTY OF SOCIAL SCIENCES
KOGNITYWISTYKA

Gracjan Popiółkowski

Sentiment Analysis Algorithms for Financial Opinions

Poznań 2024

Contents

Introduction

In this report, I present a comparison of the results of financial sentiment analysis. I will use an artificial intelligence model and compare how it performs with different parameters in classifying whether a given sentence has a positive, neutral, or negative sentiment.

CHAPTER 1

Data Presentation

1.1. Dataset

I used the "Financial Sentiment Analysis" dataset provided by user sbhatti on the Kaggle platform. The dataset includes over 5 financial opinions, classified into 3 sentiment categories - positive, neutral, or negative.

1.2. Data Preparation

I applied a series of preprocessing operations to the text to enhance the model's learning efficiency. These operations include:

- Tokenization: breaking sentences into words.
- Lowercasing: standardizing the text by converting all characters to lowercase.
- Punctuation removal: eliminating punctuation marks from the text.
- Stop words removal: excluding words that do not carry significant semantic meaning for the analysis.
- Lemmatization: reducing words to their base or dictionary form.
- Stemming: reducing words to their root form using a stemming algorithm.

CHAPTER 2

Sentiment Analysis Algorithms

Text sentiment analysis algorithms are artificial intelligence and natural language processing (NLP) methods that enable machines to understand, interpret, and classify subjective information contained in textual data. The goal of these algorithms is to automatically assess human-written text to determine the emotional tone, attitude, or feelings of the author towards the subject being discussed. Sentiment analysis is used in various fields such as marketing, customer service, social media monitoring, finance, and many others, providing valuable insights into people's opinions, moods, and preferences.

In the following sections, I will present a comparison of sentiment classification results after modifying different parameters.

CHAPTER 3

Results Analysis

3.1. The Simplest Network Model

`Model 1` - this is the simplest neural network model, defined using the Keras API in TensorFlow. It is an example model for text sentiment analysis, utilizing three main layers: Embedding, GRU, and Dense.

- `Embedding Layer` - This layer transforms word indices into dense vectors of fixed length (10 in our case). The maximum number of unique words that the model can process is set to 50,000. The dimensionality of the vector space into which the words will be embedded is set to 10.
- `GRU(16)` - GRU is a type of recurrent layer that helps the model remember long-term dependencies in sequential data through gating mechanisms. The number 16 refers to the number of units (neurons) in the GRU layer.
- `Dense Layer` - This is a classic fully connected (dense) layer that generates the output probability distribution for the three classes (hence 3). It uses the softmax activation function to convert output values into probabilities, allowing the model to classify text into one of three sentiment categories (positive, negative, neutral).
- `Compilation` - The model is compiled with the Adam optimizer, the sparse_categorical_crossentropy loss function (suitable for multi-class classification where labels are integers), and the accuracy metric to track classification accuracy.

I will compare the models based on `accuracy` and `val_accuracy`, which evaluate classification performance on the training and validation (test) datasets. `Val_accuracy` measures how well the model generalizes to new, unseen data. It is important to check whether the model is overfitting, which can occur when training accuracy is high, but validation accuracy is lower. The results for the baseline model:

- `Accuracy` = 0.9048
- `Val_accuracy` = 0.6544

Thus, the classification accuracy on new data is 65.44

3.2. Comparison of 5 Models with Modified Parameters

In the second part of the analysis, I will present the results of 5 models in which I changed one key parameter from the baseline model.

3.2.1. Model 2 - Increased Embedding Size

In Model 2, I decided to increase the embedding size from 10 to 50, meaning that each word will be represented by a vector with more dimensions. Theoretically, this allows for richer semantic representations of words, capturing more complex relationships between them, and better distinguishing between words with different meanings. The performance of this model was:

- `Accuracy` = 0.9180
- `Val_accuracy` = 0.6270

This model performed slightly better on the training data but worse on the test data.

3.2.2. Model 3 - Adding Dropout Layers

Next, I introduced a dropout function (with a value of 0.5) to the baseline model to see if it could improve results by reducing overfitting. The results are:

- `Accuracy` = 0.9155
- `Val_accuracy` = 0.6416

3.2.3. Model 4 - Using LSTM Instead of GRU

Switching from GRU to LSTM may improve the model's ability to capture long-term dependencies in the data, but it also increases computational complexity and the risk of overfitting. The results are:

- `Accuracy` = 0.9150
- `Val_accuracy` = 0.6493

3.2.4. Model 5 - Increasing Model Depth

Adding additional layers to the model (increasing its depth) can capture more complex and abstract representations of the data. In theory, deeper models have a greater capacity to learn intricate patterns, which may lead to better performance. The results are:

- `Accuracy` = 0.9163
- `Val_accuracy` = 0.6390

3.2.5. Model 6 - Applying a Convolutional Neural Network

Finally, I added a convolutional layer. CNNs are traditionally used in image processing but have proven effective in NLP tasks as well, due to their ability to detect local patterns in data.

- `Accuracy` = 0.9142
- `Val_accuracy` = 0.6356

3.3. Comparison of All 6 Models

In the third part of my analysis, we will examine how the results of all six models compare in terms of accuracy:

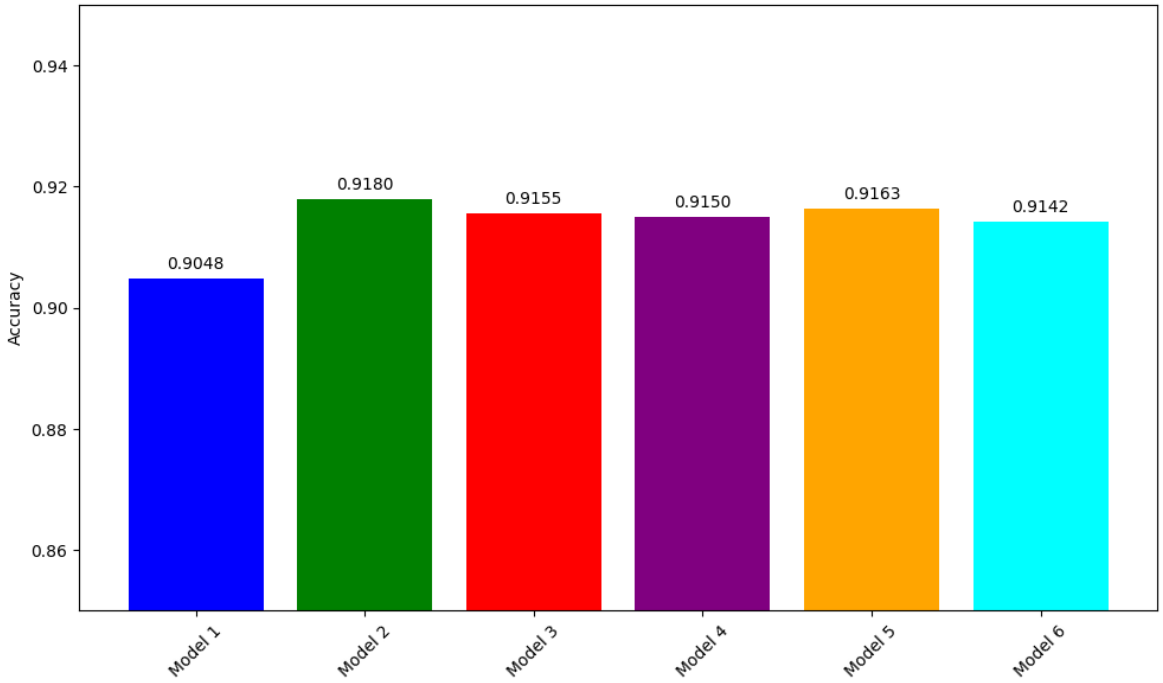

Figure 3.1: Comparison of all 6 models in terms of accuracy.

and in terms of val_accuracy:

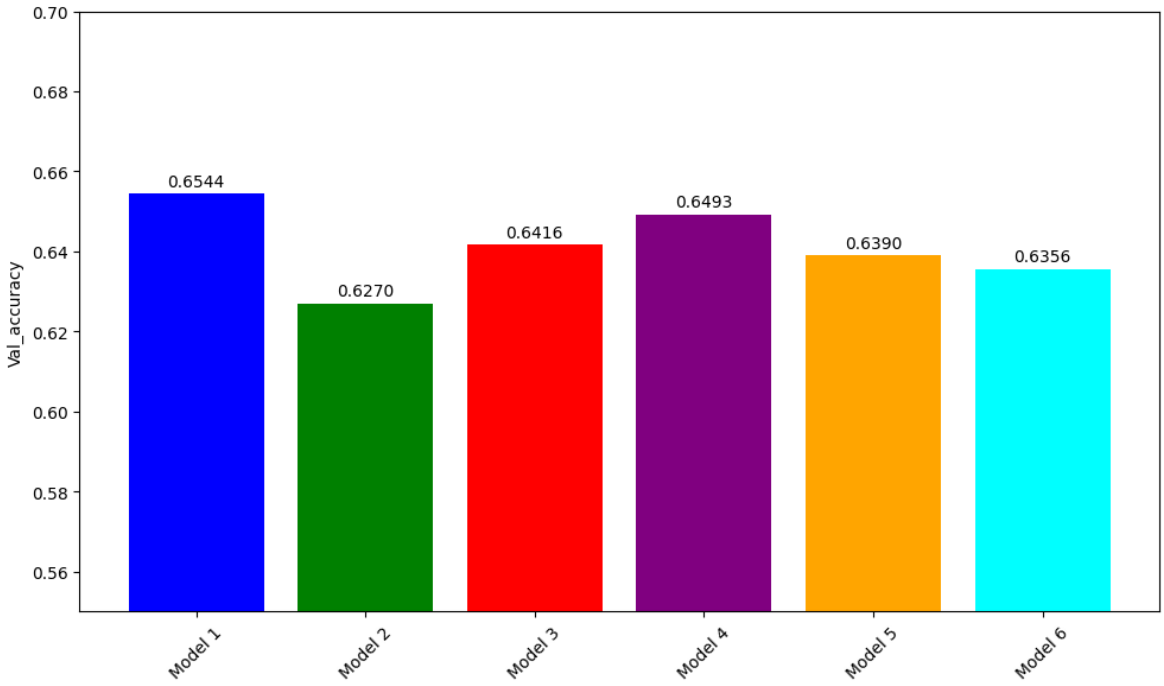

Figure 3.2: Comparison of all 6 models in terms of val_accuracy.

As we can see, in terms of test set accuracy, I was not able to find a better solution than the baseline Model 1. All models from 2 to 6 performed better on the training sets by approximately 1 percentage point.

Summary

In conclusion, in my work, I used various neural network models for sentiment classification, and the baseline model turned out to be the best in terms of accuracy:

- `Accuracy` = 0.9048
- `Val_accuracy` = 0.6544

The best result in val_accuracy was achieved by Model 2 with an increased embedding size:

- `Accuracy` = 0.9180
- `Val_accuracy` = 0.6270

The constructed models achieved approximately 65% accuracy in classifying sentiment in sentences from the test set. With three possible classes—positive, neutral, and negative—the chance of randomly assigning the correct sentiment is 33.3%. Therefore, my model has almost double the chance of correct classification. However, further tests would certainly be beneficial to determine whether such an artificial intelligence algorithm can achieve even better results. Additionally, a larger dataset would likely improve the model's effectiveness.