Data Mining II - Final Project

Sarcasm Detection in News Headlines

A picture containing newspaper

Description automatically generated

Presented by:

Anjali Shalimar

Ayshwarya Srinivasan

Digvijay Kawale

Vivek Sahoo

**Index**

**Executive Summary**

1. Why Sarcasm Detection?

2. Data Collection

3. Data Cleaning

4. Training and Testing Data Split

5. Exploratory Data Analysis

6. Modeling

* Pre-Data Processing
* Building Keras Model
* Model 1 – Base Model
* Model 2 – Post Padding
* Model 3 – Sans Lemmatization
* Model 4 – Including Stop Words
* Hyperparameter Tuning
* Results from Hyperparameter Tuning
* Final Model

7. References

**Executive Summary**

**Goal**

The goal of this project is to detect the sarcasm in the news headlines using the machine learning approach of Natural Language Processing. The aim is to build a tool to classify news headlines into Sarcastic or Non-Sarcastic Headlines.

**Approach**

We will be using Natural Language Processing to build a model that classifies the news headlines into Sarcastic or Non-Sarcastic. The standard process of Natural Language Processing consisting of Tokenizing, Removing Stopwords, Stemming and Lemmatization, Bag of Words and N-grams has been applied before building the model. As the sequence of word use of important for Sarcasm Detection, we will be using the Recurrent Neural Network to get best results out of built model. Further, we know that the Recurrent Neural Network doesn’t have memory, hence the weights for initial values of input cannot be adjusted as much as the latest values, due to vanishing Gradient Descent. To tackle this, we will be using LSTM – Long Term Short Memory. The final model will be evaluated based on its accuracy of detecting sarcasm in news headline. The original data will be split into 70% training and 30% testing so that we will be able to choose the best model based on predicting the Sarcasm in testing dataset.

**Results:**

The final model we built is a Keras model with LSTM based on the training and testing accuracy. The training accuracy obtained by this model is 91.9% and the testing accuracy obtained is 85.6%.

**1. Why Sarcasm Detection?**

Sarcasm has always been hard to detect. It is difficult to detect for few people even in their normal conversations. Sarcasm is a persuasive linguistic phenomenon in online documents that express subjective and deeply felt opinions. Detection of sarcasm is of great importance and beneficial to many NLP applications, such as sentiment analysis, opinion mining and advertising. The other important application is the conversational AI, as in its current stage it has not completely emulated to human conversations. Different conversational AI systems like ‘Siri’ in Apple devices, ‘Google assistant’ in Google devices, ‘Alexa’ – conversational music station of Amazon, etc. are good at doing simple tasks like setting alarms and reminders but they lack the real feel of human conversation. Specifically, they are very weak at detecting the sarcasm and thus making it a topic of interest for further development of such products.

With this motivation, we are attempting to explore this space by doing a project on a topic ‘Sarcasm Detection in News Headlines’. Sarcasm writing is common in news headlines, which helps to build an environment for us to study one specific application. Also, a lot of past studies in sarcasm detection have used Twitter datasets collected using hashtag-based supervision but such datasets are noisy in terms of labels and language. Furthermore, many tweets are replies to other tweets and detecting sarcasm in these requires the availability of contextual tweets. Thus, to overcome this limitation related to noise in Twitter datasets, news headlines data proves to more concise and direct application to our topic of interest. So, we decided to take this as our final project topic and see how different machine learning techniques can be helpful to achieve our aim.

**2. Data Collection**

The data for news headlines was sourced from Kaggle. The dataset contains news headlines collected from two news sources – The Onion and HuffPost. The Onion is known for sarcastic headlines, thus becomes a good source of data for this project. The headlines collected from HuffPost are non-sarcastic headlines. Using a dataset with a proper mix of sarcastic and non-sarcastic headlines will be helpful to build non overfitting models. The data from the Kaggle is in the format shown below:

|  |  |
| --- | --- |
| **Column Name** | **Column Description** |
| is\_sarcastic | Flag to classify headlines into sarcastic and non-sarcastic headlines. It has value 1 for sarcastic headlines and 0 for non-sarcastic headlines. |
| headline | Headline of the news article. |
| article\_link | Link to the original news article. Useful in collecting supplementary data. |

**Table 2.1: Data Dictionary**

Data contains a total of 26709 observations. Below is the snapshot of the sample data:

A screenshot of a cell phone

Description automatically generated

**Fig 2.1: Sample data**

**3. Data Cleaning**

Data cleaning for this data set involves the standard steps of Natural Language processing. The data cleaning steps involved are outlined below:

* **Removing Punctuations**: All the punctuations marks are initially removed from both the sarcastic and non-sarcastic headlines. In natural language processing words have more importance than the punctuations. They are removed to keep our main focus on different words used in headlines.
* **Tokenization**: Tokenization is a process of breaking a stream of text up words, phrases, symbols and other meaningful elements called tokens. Typically, we convert natural language types into a logical entity for programming languages by tokenizing. Different words from both sarcastic and non-sarcastic headlines are tokenized using the ‘nltk’ package in python.
* **Removing Stopwords**: A stopword is basically commonly used works such as ‘a’, ‘an’, ‘the’, etc. which do not have any sentiment associated with them and will not contribute in our modeling process. Thus, we remove all the stopwords from both sarcastic and non-sarcastic headlines.
* **Creating a single lists of Sarcastic Headline words and non-sarcastic headline words**: After the preprocessing done in the previous steps of data cleaning, we will save all the remaining words in single lists of sarcastic headline words and non-sarcastic headline words to help us in further analysis.
* **Lemmatization**: Lemma is basically the basic form of word. For example, the words studied, studies, studying has its basic form as study. Thus, we will be converting all the words in both the lists to their lemmas.
* **Bag of words**: A bag of words in a simplifying representation used in Natural Language processing aiming at extracting the features from the words. Thus, we will be making a bag of words for sarcastic headlines and one for non-sarcastic words.

**4. Training and Testing Data Split:**

The data was split into 70% training and 30% testing sets. Training data set has 18696 observations and testing data set has 8013 observations. The seed used for training testing split is 15782. All the exploratory data analysis and model fitting will be done use the training data set and testing data set will be used to evaluate the model performance.

**5. Exploratory Data Analysis**

* **Word count comparison**: Below bar graph shows the most frequent occulting words in news headlines.

A screenshot of a cell phone

Description automatically generated

**Fig. 5.1 Frequency distribution of most commonly occurring words in the headlines**

* **Proportion of Sarcastic vs non-Sarcastic:** 44% of headlines are sarcastic while the remaining 56% are non-sarcastic. Figure 5.2 shows the pie chart for the proportions of sarcastic and non-sarcastic headline.
* **Word clouds for Sarcastic and non-Sarcastic news headlines:**  A word cloud is an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance. Below are the word clouds for sarcastic and non-sarcastic news headlines. We can see that man, new, area, report the most frequently used word in sarcastic headlines and words like trump, year, new Donald are the most frequently words used in non-sarcastic news headlines.

**A close up of a device

Description automatically generated**

**Fig. 5.2 Proportion of Sarcastic and Other Comments**

**A picture containing food

Description automatically generated**

**Fig. 5.3 Word Cloud – Sarcastic Headlines**

A close up of text on a black background

Description automatically generated

**Fig. 5.4 Word Cloud – Non-Sarcastic Headlines**

**6. Modelling:**

We now move towards the modeling part of the project with an aim to build a tool that is able is detect sarcasm in sentences. We will be processing the data to transform it as per our model input requirement. Below is the pre-data processing for modelling purpose.

**Pre-data Processing:**

* Removing punctuations: Most text contains punctuations. In detecting sarcasm, the presence of punctuations doesn't necessarily contribute to the model performing better. So, we aim to strip the data of all punctuations.
* Remove digits: We are going to vectorize our data and convert the strings to numbers. Presence of numbers in the data would not help in identifying the tone any better. Moreover, the pre-existing digits might interfere with the vectorization process. Hence all numbers are removed as well.
* Converting to lower case: Converting the text to lower case helps make the data uniform.
* Removing stop words: Most headlines or any natural language data would contain stop words that are usually removed as stop words generally appear in abundance and do not provide any valuable information during classification.
* Lemmatization: Lemmatization is the process by which any inflected version of a word is converted to its base word so that all forms of a word are treated the same.
* Vectorization and padding: Vectorization are a process by which words are mapped to the numeric vectors. For LSTM model, the input should be of same size. Hence, we pad the vectors with zeros to ensure uniformity.

**Building Keras Model**

We will be building a Keras model for the data set. Below are the steps involved in the Keras model building:

* **Embedding layer**: The Embedding layer is used to create word vectors for incoming words. It sits between the input and the LSTM layer, i.e. the output of the Embedding layer is the input to the LSTM layer.
* **LSTM Layer**: The LSTM transforms the vector sequence into a single vector containing information about the entire sequence.
* **Intermediate Layer:** There is a Dense intermediate layer with 64 neurons and with activation function **relu**.
* **Output Layer**: The final output we want from this model is whether the headline is sarcastic or not. So, we want to perform classification. The output layer's activation function is thus **sigmoid.**

**Model 1 - Base Model**

We will first be building a base model with the following processing:

* Remove punctuations, digits and convert the text to lowercase.
* Remove all stopwords
* Perform Lemmatization
* Perform tokenization and pad the resulting sequence – prepadding

It was observed that we were getting an accuracy of 84.7% using this base model.

**Model 2 – Post padding**

We first build a model very similar to the base model with the following change: After performing the tokenization, do post-padding.

**Example:** Before padding: [234,5,67,12]

The maximum length is 7

Pre-padding: [0,0,0,0,234,5,67,12]

Post-padding: [234,5,67,12,0,0,0,0]

Surprisingly, we see a huge drop in the accuracy from 85% to 55% now. The reason why this is because we are building a Long Short-term memory model. When the padding is in the beginning, the useful content is at the back and is therefore the latest information the model takes in. This stays in memory and results in a better model. We are going to proceed further with pre-padded sequence for future models.

**Model 3 – Sans Lemmatization**

We now build a model that is a modification of our base model. We want to see the effect of lemmatization. Lemmatization is the process by which any inflected version of a word is converted to its base word so that all forms of a word are treated the same. We want to see if in detecting sarcasm, the **effect of inflect** plays a role in improving the efficiency of our model.

We observed that not performing lemmatization improves the accuracy but just slightly from 85% to 85.6%

**Model 4 – Including Stop Words**

We test the effect of what if stop words are on the model. In general, in NLP models, we remove stop words. But our theory is that the stop words might actually have an effect in identifying the sarcasm in a sentence.

We want to see if in detecting sarcasm, the effect of stop words plays a role in improving the efficiency of our model.

As per the results after applying this, the accuracy did not increase too much from the base model.

**Hyperparameter Tuning**

Hyperparameter optimization (or tuning) is the process of choosing a set of optimal parameters for a machine learning algorithm. Data preprocessors, optimizers and ML algorithms all receive a set of parameters that guide their behavior. To achieve optimal performance, they need to be tuned to fit the statistical properties, type of features and size of the dataset used. The most typical hyperparameters in deep learning include learning rate, number of hidden layers in a deep neural network, batch size, dropout, etc. In NLP we also encounter a number of other hyperparameters often to do with preprocessing and text embedding such as type of embedding, embedding dimension, number of RNN layers, etc. The hyperparameters in consideration are listed below. We will be using the grid search method for selecting the best parameters.

* Layer Activation
* Number of Epochs
* Optimizer

**Results from Hyperparameter Tuning:**

From the grid search we get the following results:

1. The best optimizer for our model is **adam**
2. The best activation to use is **relu**

We also try various epoch values: 2,5,10. Below table shows the results by using different epochs.

|  |  |  |
| --- | --- | --- |
| **Epoch Value** | **Training Accuracy** | **Testing Accuracy** |
| 2 | 91.9% | 85.6% |
| 5 | 98.7% | 82.8% |
| 10 | 99.3% | 83.3% |

**Table 6.1: Training and Testing Accuracy for different Epochs**

We see that for epochs higher than 2, even if the training accuracy increases the testing accuracy goes down. This could be because of overfitting.

**Final Model**

The final model we build is a Keras model with LSTM based on the training and testing accuracy.

Preprocessing: Removing stop words, punctuations, digits and converting to lower case.

Epochs: 2

Activation: relu

Output Activation: sigmoid

Optimizer: adam

Training accuracy: 91.9%

Testing accuracy: 85.6%

**7. References:**

<https://link.springer.com/chapter/10.1007/978-3-319-08010-9_49>

<https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection>

<https://keras.io/getting-started/sequential-model-guide/>

<https://towardsdatascience.com/why-you-should-avoid-removing-stopwords-aa7a353d2a52>

<https://towardsdatascience.com/how-to-beat-automl-hyperparameter-optimisation-with-flair-3b2f5092d9f5>