NEWS CLASSIFIER

ABSTRACT

Here is a Neural Network (Natural Language Processing) classifier. Our document classification model for binary classification was successful with 85% + accuracy.

2019

BACKGROUND.

In brief, our approach consists of two stages.

Stage One – Develop a classifier to automate the analysis and subsequent segregation of textual content from inputted news articles into 'Agriculture' or 'Non-Agriculture' categories. Since Phase I deals exclusively with Agricultural datasets in the English medium, the input is limited to English news articles related to Agriculture at this stage.

Stage Two – Develop a classifier to further categorize the 'Agriculture' articles based on their level of data-intensity. For our purpose, there can be three such categories, as follows.

1. Non data-intensive – Articles which do not quote any data and are simply based on reporting of facts or circumstantial details are considered non data-intensive.
2. Anecdotal – Articles which present preliminary facts or figures (simple percentages, areas, prices, population figures, etc.) without citing credible sources are considered anecdotal articles.
3. Data-intensive – Articles which contain substantial data in-text (trends, comparisons, etc.) or in the form of visualizations (graphs, charts, infographics, etc.) are considered data-intensive.

The details of the methodology we followed to implement this approach are presented in the following report.

PROBLEM STATEMENT

Develop a convolutional neural network classifier that automates the segregation of inputted news articles into 'Agriculture' or 'Non-Agriculture' categories.

Note – We were unable to complete the implementation of Stage Two of our approach due to our short engagement period. However, the methodology followed in Stage One (see below) is directly transposable to the problem statement for Stage Two as well.

THEORY

For the purpose of our project, we have used a neural network model for natural language processing. Neural Networks are basically a family of powerful machine learning models. Natural language processing (NLP) on the other hand is the field of designing methods and algorithms that take as input or produce as output unstructured, natural language data.

Specifically, we are using supervised machine learning algorithms that attempt to infer usage patterns and regularities from a set of pre-annotated input and output pairs. Consider, for example, the task of classifying a document into one of two categories: agriculture articles and non-agriculture articles. Obviously, the words in the documents provide very strong hints, but which words provide what hints? Writing up exhaustive rules for this task is rather challenging. However, readers like us can easily categorize a document into its topic, and

then, based on a few hundred human-categorized examples in each category, we let a supervised machine learning algorithm come up with the patterns of word usage that help categorize the documents. Machine learning methods excel at problem domains where a good set of rules is very hard to define but annotating the expected output for a given input is relatively simple.

Given a large set of desired input-output mapping, neural networks work by feeding the data into a network that produces successive transformations of the input data until a final transformation predicts the output. The transformations produced by the network are learned from the given input-output mappings, such that each transformation makes it easier to relate the data to the desired label.

A major component in neural networks for language is the use of an embedding layer, a mapping of discrete symbols to continuous vectors in a relatively low dimensional space. When embedding words, they transform from being isolated distinct symbols into mathematical objects that can be operated on. The representation of words as vectors is learned by the network as part of the training process. Going up the hierarchy, the network also learns to combine word vectors in a way that is useful for prediction.

A major kind of neural network architecture is **Convolutional feed-forward networks**. This is the network **we use** in this project. They are specialized architectures that excel at extracting patterns in the data: they are fed arbitrarily sized inputs and are capable of extracting meaningful local patterns that are sensitive to word order, regardless of where they appear in the input. These work very well for identifying indicative phrases or idioms of up to a fixed length in long sentences or documents.

As the name suggests, neural networks were inspired by the brain's computation mechanism, which consists of computation units called neurons. While the connections between artificial neural networks and the brain are in fact rather slim, we repeat the metaphor here for completeness. In the metaphor, a neuron is a computational unit that has scalar inputs and outputs. Each input has an associated weight. The neuron multiplies each input by its weight, and then sums them, applies a nonlinear function to the result, and passes it to its output. The following figure shows such a neuron.

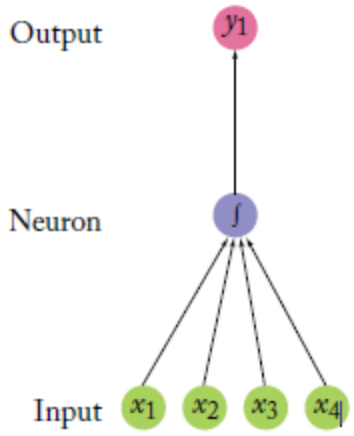


Figure 1: A single neuron with four inputs

These neurons are connected to each other, forming a network: the output of a neuron may feed into the inputs of one or more neurons. Such networks were shown to be very capable computational devices.

A typical feed-forward neural network may be drawn as in Figure 2. Each circle is a neuron, with incoming arrows being the neuron's inputs and outgoing arrows being the neuron's outputs. Each arrow carries a weight, reflecting its importance (not shown). Neurons are arranged in layers, reflecting the flow of information. The bottom layer has no incoming arrows and is the input to the network. The top-most layer has no outgoing arrows and is the output of the network. The other layers are considered 'hidden'. The sigmoid shape inside the neurons in the middle layers represent a nonlinear function (i.e., the logistic function [1] 1/1+e^(-x))[1] that is applied to the neuron's value before passing it to the output. In the figure, each neuron is connected to all of the neurons in the next layer—this is called a *fully connected layer* or an *affine layer*.

Now, coming to loss functions. When training a neural network, much like when training a linear classifier, one defines a loss function L (ŷ, y), stating the loss of predicting ŷ when the true output is y. The training objective is then to minimize the loss across the different training examples. The loss L (ŷ, y) assigns a numerical score (a scalar) to the network's output ŷ given the true expected output y. Loss function is used to measure the performance of the model and inconsistency between actual y and predicted value ŷ . Performance of model increases with the decrease value of loss function.

1 A logistic function or logistic curve is a common "S" shape (sigmoid curve) curve that puts a limit on growth instead of unchecked growth like in case of an exponential function.

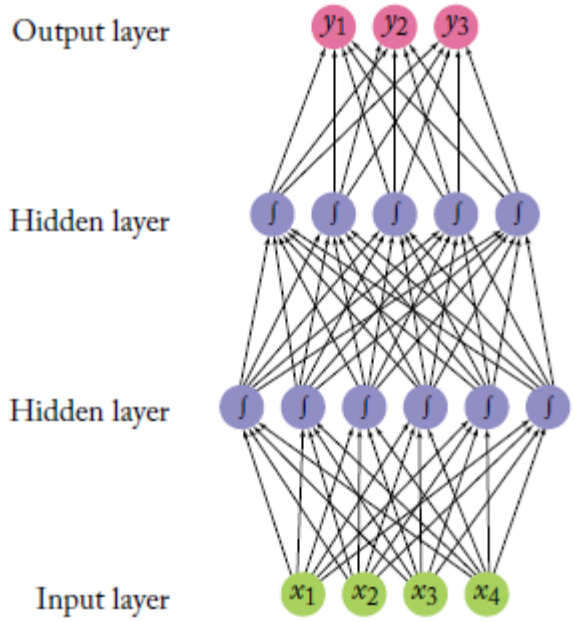


Figure 2: Feed-forward network with two hidden layers

Once our basic model has been built with all hyper-parameters specified, we run our training dataset on it. Now, the first step in ensuring that our neural network performs well on the testing data is to verify that our neural network does not overfit. Overfitting happens when our model starts to memorize values from the training data instead of learning from them. To identify if our model is overfitting, we can just cross check the training accuracy and testing accuracy. If training accuracy is much higher than testing accuracy, then one can posit that our model has overfitted. We can also plot the predicted points on a graph to verify. There are some techniques to avoid overfitting:

- Regularization of data (L1 or L2).
- Dropouts — Randomly dropping connections between neurons, forcing the network to find new paths and generalise.
- Early Stopping — Precipitates the training of the neural network, leading to reduction in error in the test set.

Next step would be hyper-parameter tuning. Hyper-parameters are values that you must initialise to the network, these values can't be learned by the network while training. In a convolutional neural network, some of the hyper-parameters are kernel size, the number of layers in the neural network, activation function, loss function, optimizer used (gradient descent, RMSprop), batch size, number of epochs to train etc. There is no direct method to identify the best set of hyper-parameter for each neural network so it is mostly obtained through trial and error. But there are some best practices for some hyper-parameters which are mentioned below,

1. Learning Rate — Choosing an optimum learning rate is important as it decides whether our network converges to the global minima or not. Selecting a high learning rate almost never gets us to the global minima as we have a very good chance of overshooting it. Therefore, we are always around the global minima but never converge to it. Selecting a small learning rate can help a neural network converge to the global minima but it takes a huge amount of time. Therefore, we have to train the network for a longer period of time.

2. Network Architecture — There is no standard architecture that gives you high accuracy in all test cases. We have to experiment, try out different architectures, obtain inference from the result and try again. One idea that is to use proven architectures instead of building one of our own.

3. Optimizers and Loss function — There is a myriad of options available for us to choose from. In fact, one could even define our custom loss function if necessary. But the commonly used optimizers are RMSprop, Stochastic Gradient Descent and Adam. These optimizers seem to work for most of the use cases.

4. Batch Size & Number of Epochs — Again, there is no standard value for batch size and epochs that works for all use cases. We have to experiment and try out different ones. In general practice, batch size values are set as either 8, 16, 32… The number of epochs depends on the developer's preference and the computing power he/she has.

5. Activation Function — Activation functions map the non-linear functional inputs to the outputs. Activation functions are highly important and choosing the right activation function helps your model to learn better. We have used the sigmoid activation function.

After performing all of the techniques above, if our model still doesn't perform better in our test dataset, it could be ascribed to the lack of training data.

We have attached our model in the appendix. Our aim now is to build a good training set to get the maximum possible accuracy percentage and identify causes behind us not reaching above 95%, if that is the case.

METHODOLOGY

STEP 1: Getting the data

We followed two steps to get the content of different news articles as a list in excel spreadsheet format. The first step was to extract the URLs of different news articles and the second step was to extract the main body of each of the articles, the URLs of which we had already extracted.

1. Getting the URLs:

To get the URLs of different news articles, we used a tool called 'Data Miner'. Data Miner is **a chrome extension** software that assists in extracting data that you see in your browser and save it into an Excel spreadsheet file.

In the data miner software, we create something called a 'recipe'. Recipes are data extraction instructions that Data Miner uses to extract data from websites. Recipes contain name and position of HTML elements on a web page.

We used one of data miner's public recipes called 'Get all links'. Using this, we were able to get all the URLs present on a web page.

2. Getting the news article content:

After removing unnecessary (advertisements, home page, about page, contact page etc.) and broken URLs from our excel spreadsheet of URLs, we moved on to the next step of extracting the content of these news articles.

We used one of python's libraries beautiful soup for this. The code has been attached in the appendix for reference.

However, following these two steps also came with its fair share of problems.

-We first started out with a spreadsheet of 1500 news article URLs and text files of the source code provided to us by Ms Ankita. We then ran an R code on these text files to extract just the main body of the news articles. But after going through these articles, we realised that there were very few agriculture articles in this set.

- Then, in order to get more agriculture articles, we extracted 2200 articles from Newsrack with the tag 'agriculture'. However, to our dismay, Newsrack proved to be a very unreliable sorter. Again, we found very few agriculture articles.

- We then turned to google news. We extracted 2000 articles from google news with the tag 'agriculture' for the period 2003-2019. From this, we found 1000 agriculture articles after manual sorting.

After these three steps, we had managed to secure at least 1000 articles of both categories (agriculture and non-agriculture) to serve as the training set for our classifier.

-Upon recognising the need for a bigger set at a later stage, we extracted about 1300 more articles. However, by this time, we had exhausted the most common search engines and therefore, visited independent news websites (The Hindu, Hindustan Times, Hitavada, Indian Express and Krishi Jagran). We are able to get a mixed set of 1300 articles which we then used to re-train our classifier.

STEP 2: Cleaning the data

In order to ensure that our data was correct, consistent and useable, our next step was to identify any errors or corruptions in the data.

Luckily our code gave us fairly clean news articles with few errors. In order to make our data set completely error-free, we first tried to identify if there was any pattern in the errors present. We found that most of the foreign symbols present in our news articles followed a systematic pattern throughout the entire set. For instance, a colon (') was always replaced

by a A hat (Â). Similarly, we found all such patterns and removed them using excel's find and replace function.

Next, we removed all duplicates from our data set realising that there might be overlapping news articles as our source was just different search engines/sorters for the first 2000 articles.

Finally, while categorizing these articles into our two categories, we manually removed any inconsistencies found in our data set such as image descriptions, irrelevant placeholders and footnotes etc.

STEP 3: Categorizing

After cleaning the entire dataset, we generated a supervised training set of 2000 *labelled articles*, i.e. 1000 in each category, agriculture and non-agriculture.

The classification of articles into agriculture and non-agriculture was done based on our *existing knowledge* and perception of what should be considered an article addressing agricultural issues. A formal list of specific rules to be followed for classification was not created at this stage. This was done intentionally to feed the classifier multiple perspectives such that it generalizes well on unseen data. This was done to avoid the problem of overfitting in the model, considering the sample size.

To ensure that we build a fair training set that allows the model to accurately learn and predict the required class, we selected 1000 agriculture and non-agriculture articles each. Labels are predefined classes that our model will predict, in our case, we have labelled agricultural articles to be 1 and the non-agricultural ones as 0.

80% of the labelled articles (1600) were used to train the classifier. The remaining 20% (400) was used as the testing set.

The parameters of the classifier were as follows:

- Vocabulary Size: This indicates the number of words in our dictionary, which was 3000.
- Maximum length: This function indicates the maximum length upto which the classifier will read the input. Since our average article size was 500 words, we have set the maximum length to 300.
- Embedding layer: Embedding implies representation of text in a manner such that words that have the similar meaning are represented similarly. It represents words in a coordinate system, where related words are placed closer together. In our network, we had 64 embedding dimensions.
- Both trunc type (remove words pre/post the maximum length) and padding type (insert missing values pre/post the text to match the maximum length) have been set to 'post'.
- Words not defined in the dictionary shall be read by the classifier as "OOV".

- Filter: A convolution 1D filter is used to map internal features of the sequence and to reduce the size of the input matrix.
- The activation functions (mapping input to output via a non-linear transform function) used are “relu” and “sigmoid”. Sigmoid function parses a value to be between 0 and 1. ReLu allows the output to take a value of 0 if it is negative, otherwise the value remains the same.
- The model is finally compiled using a loss function (loss='binary_crossentropy'), an optimizer (‘adam') and metrics ('accuracy').

We then train the model and *cross-validate* on our test set.

In the first round of testing, our model predicts with approximately ***85% accuracy***.

STEP 4: Revalidation

The predictions made by the model on the test set were revalidated manually to measure the accuracy of the classifier and make further improvements in the model.

Considering the errors existing in the network, we decided to retrain the model by feeding more labelled articles. However, for retraining, a specific list of rules was discussed and agreed upon, on the basis of which all further classification was carried out.

STEP 5: Retraining

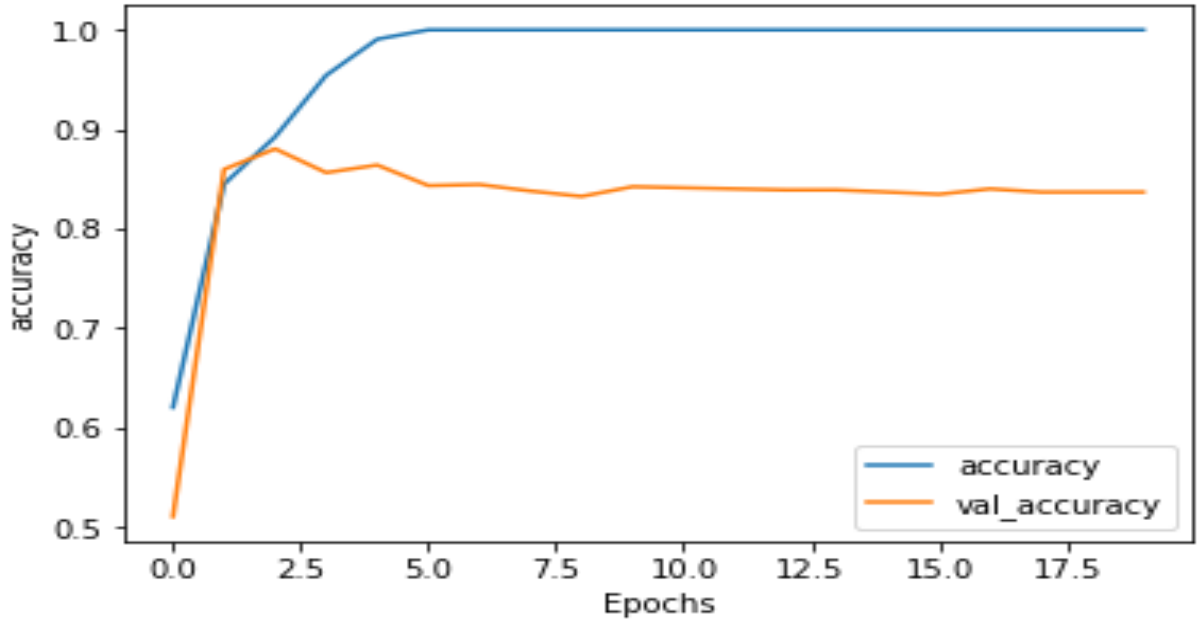
1300 articles were further added in the existing corpus of 2000. All the articles were reclassified according to the following rules:

- All articles having agriculture as their ***main theme*** have been labelled 1 (for example articles about farm loan waivers, farmer suicides etc).
- All Allied sectors alongside agriculture have been labelled 1, these comprise, fishery/livestock/horticulture/forestry and logging/agricultural products/irrigation/poultry/dairy).
- All articles discussing environment and sustainability with agriculture as its main theme have been labelled 1 (organic farming, sustainable agriculture, etc).
- All articles covering any advancements in Agricultural Research have been labelled 1. These include invention of sophisticated farm tools and equipment, as well as building of apps facilitating agricultural activity.
- All articles discussing GM crops (genetically modified) have been labelled 1.
- All articles mentioning climate, water crisis, monsoon and its impact on *agriculture as the major theme* have been labelled 1.
- All articles mentioning protests and legal disputes by/involving farmers have been labelled 1.
- All individual farmer stories, enlisting issues they are facing or inventions they have discovered at a micro/personal level have been labelled 1.

- All articles describing agricultural events/conferences have been labelled 1.
- All articles describing admissions in agricultural degrees or courses have been labelled 0. However, those describing the content of their curricula or research undertaken at agricultural universities have been labelled 1.
- All articles discussing the dietary or nutritional aspects of crops have been labelled 0.
- All articles touching upon agriculture briefly, i.e. as a minor theme have been labelled 0.

INFERENCES

In the first round of testing, our model yielded a prediction accuracy rate of 85%. This is depicted in the following graph, which plots accuracy against the number of epochs. The distance between the blue trend line and orange trend line depicts the extent of error in prediction. It is this distance i.e. the prediction error that we try to minimize by experimenting with the hyper-parameters used in the model. If the orange trend line depicting validation accuracy closely traces the blue trend line, this means the chosen hyper-parameters are optimal and the model has reached an excellent rate of accuracy.



Upon retraining the model, the accuracy rate is expected to increase further. Equivalently, the loss function is expected to be minimized as we achieve higher and higher rates of accuracy.

LIMITATIONS

As with most studies, the design of the current study is subject to limitations.

- The sample size for training was initially fixed at 2000, however, post revalidation and retraining it has been increased to 3300. For Convolution Neural Network studies, the training set should be vast and exhaustive to ensure accurate predictions.
- Classification of articles has been done on the basis of rules discussed upon and decided amongst us, thus there exists an element of subjectivity. For instance, poultry and forestry might or might not be considered part of agriculture, but in this study, all allied sectors have been included.

CONCLUSION AND RECOMMENDATIONS

This study has been the **first step** in attaining the end goal of the project- examining the level of data journalism in India. This study must be taken forward, with another classifier built in continuation to this one, which would examine the quality of data used in news reports. Through this study, the classification of articles into agriculture and non-agriculture has been made possible. The next step will focus on the categorisation on basis of data-intensity. Once that is achieved, this study will enable us to understand the ongoing trends of data journalism in the field of agriculture. The same exercise can be expanded to other fields.

We set out the following recommendations –

- The existing training set needs to be expanded, allowing the model to be retrained until an exhaustive set is attained. This might also prove to bring about gains in accuracy.
- The list of rules agreed upon can be discussed and modified further so as to suit the needs of the project. However, care must be taken to avoid the problem of overfitting.
- The hyper-parameters of the model can be tweaked around to check for a higher accuracy percentage.