Homework 2 - Natural Language Processing

Piyusha Jaisinghani

1 Explain how ANN works and how multiple layers can be used for improving the accuracy.

ANN is a non-linear model inspired from the neural structure of the brain. An ANN is made of three layers- input, hidden and output each consists of neurons. These neurons hold a feature each of the input data. For a task like classification, each neuron identifies a particular feature in the input and pass it on to the next layer. For example neurons in input layer pass the information to the hidden layer neurons, which in turn gives the information to the output layer.

The figure 1 shows a basic structure of ANN. [1]

1.1 Connections between Neurons

Each neuron in a layer is connected to every other neuron in the next layer. The connection between input 1 (I1) and hidden layer neuron (H1) will have a weight W_{I1H1}, similarly we have W_{I1H2} , W_{I1H3} and so on.

1.2 Activation function

Activation function determines the firing in a neuron. It converts an input of a node in a layer to its output, this output is in turn used as an input to the next layer. Basically activation function helps in understanding complicated and non-linear complex function mappings between input and output. If the activation function is not present in the ANN, then the neural network is a linear model which is incapable of capturing and understanding complex behaviour of a huge dataset consisting of text, audio, video, images etc. [2]

1.3 Steps explaining working of ANN

- In the start weights are randomly initialized. The input is then passed to the next layer(in this case hidden layer) along with the weights.
- The activation function combined with the input to hidden layer and weights produces the output given to the output layer.
- The error at output is then used to adjust the weights of each neuron layer by layer.
- After the weights are adjusted the above process is repeated until the convergence criteria is met.

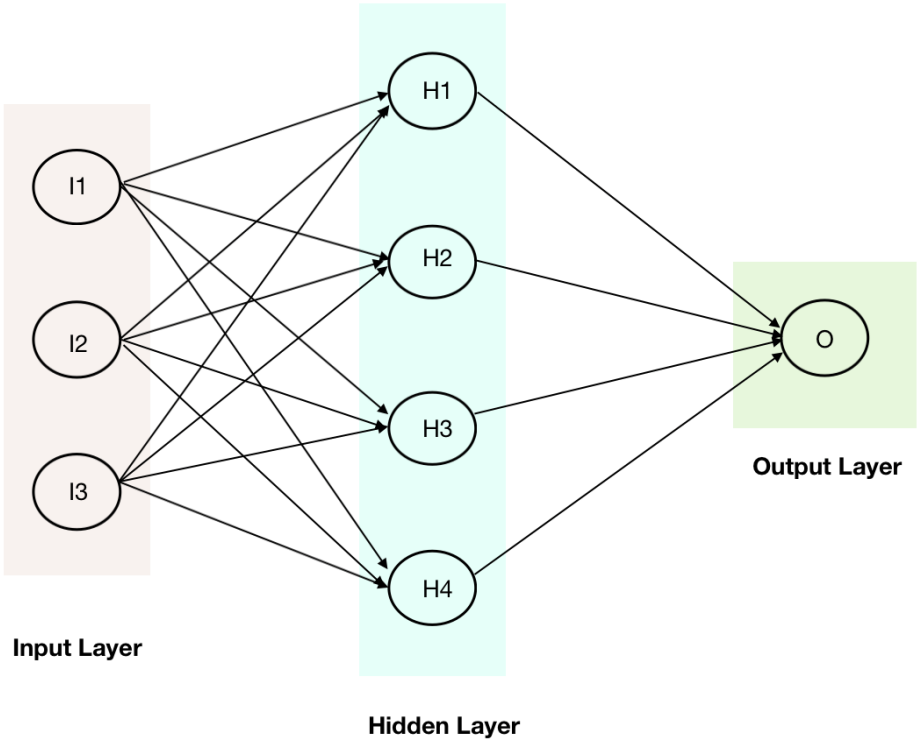

Figure 1: Structure of ANN.

1.4 Multiple layers in ANN

As discussed previously each neuron in a layer helps identify a feature in the dataset. This implies that for a complex and huge dataset it will be beneficial to add more layers that will help extract more features. Identifying multiple features in a complicated and large dataset and taking decision based on it will result to increase in the accuracy.

2 Explain ANN using matrix and sum-based representation.

The input is fed to the input layer with three neuron I1,I2,I3, represented by the matrix X, Weight matrix W. Z is the matrix representation of the dot product of X and W plus a bias.

$$Z = Wx + b$$

$$a = f(Z)$$

The activation function is applied element-wise because the output for a layer is determined by the value of each and every neuron from the previous layer and that propagates to the upcoming layer.

$$\begin{bmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \\ x41 & x42 & x43 \end{bmatrix} \begin{bmatrix} w11 & w12 & w13 & w14 \\ w21 & w22 & w23 & w24 \\ w31 & w32 & w33 & w34 \end{bmatrix} = \begin{bmatrix} z11 & z12 & z13 & z14 \\ z21 & z22 & z23 & z24 \\ z31 & z32 & z33 & z34 \\ z41 & z42 & z43 & z44 \end{bmatrix}$$

where $z11 = x11 * w11 + x12 * w21 + x13 * w31$ and so on.

The activation function is then applied to the Z calculated such that:

$$a11 = f(Z_1^{(1)})$$

$$a12 = f(Z_2^{(1)})$$

$$a13 = f(Z_3^{(1)})$$

$$a14 = f(Z_4^{(1)})$$

The output from the hidden layer is then passed as the input to the output layer where it is again multiplied by weights and after activation the output is produced.

$$Z_1(2) = a11 * W_{11}(2) + a12 * W_{12}(2) + a13 * W_{13}(2) + a14 * W_{14}(2)$$

The output O is

$$o = f(Z_2^{(2)})$$

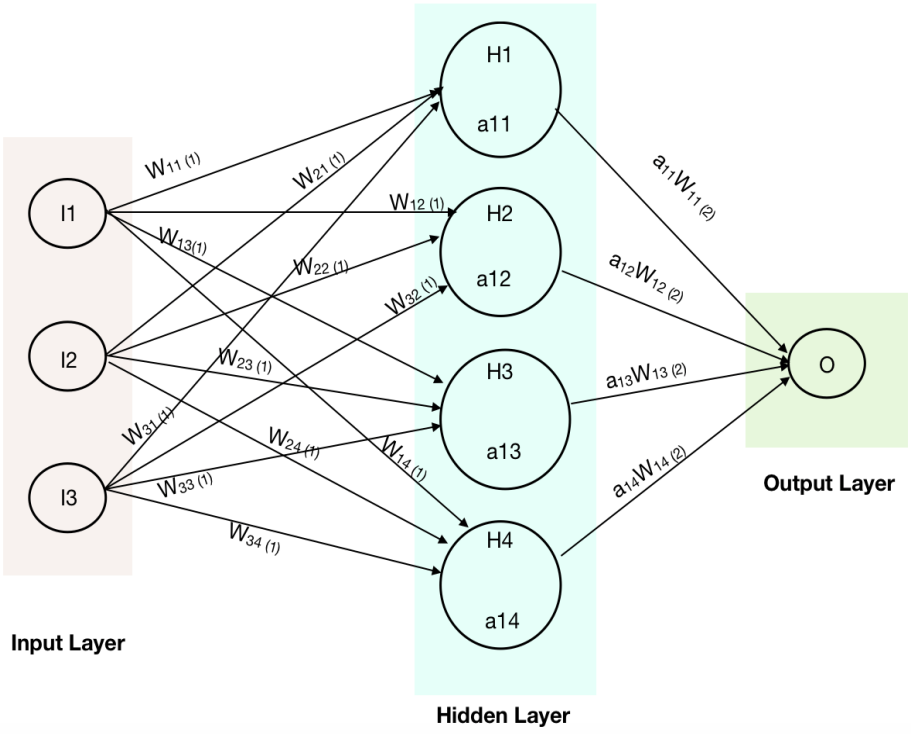

Figure 2: ANN with weights and activation function.

3 Explain what is the problem in named entities and how window classification can be used to solve the problem.

Named Entity is used for the task of classification words in the text into pre-defined categories. For example whether a word is a person[PER], Organization[ORG], location[LOC] etc. Problems with named entity are as follows:

- The named entity fails to understand the boundary of the entity for example Fifth Third Bank, in this case it is an entity organization but NER can treat it as three separate words.
- The entity of a particular word depends on it context and here NER fails to classify whether it is an entity or not. For example the White House should be an entity and a white house should not be.
- The NER works with predetermined closed list of categories which might not be fit for a new entity.
- A particular word can mean different thing in different contexts. For example Paris as in Paris Hilton or Paris in France.

To over come this issue window classification is used. Window classification is technique used to solve ambiguity word meaning problem, it helps in classifying a word's meaning/entity based on its neighbouring words. For example consider the sentence temperature in Orlando is hot. Here Orlando could be a location Orlando, Florida or Orlando could be the name of a person Orlando Pita (a famous hairdresser). In the window classification a window size is defined in which the center of the window is a word which is classified based on the context words surrounding it in the window.

$$x_{window} = [x_{temperature} x_{in} x_{Orlando} x_{is} x_{hot}]^T$$

Resulting vector is a column vector

$$x_{window} = x2R^{5d}$$

4 Explain how back propagation works and what it is used for.

In a neural network once we have received an output after a forward propagation, we compare it with the target output to get an error function or a cost function. The cost function forms the basis of back propagation. The major goal of back-propagation is to reduce this cost function and to get the actual output closer to the target output.

To achieve this we start by updating the weights in the network starting from the back side to minimize the error for each input neuron and the complete network.

Below figures explains the forward pass and then backward pass to reduce the error. [3]

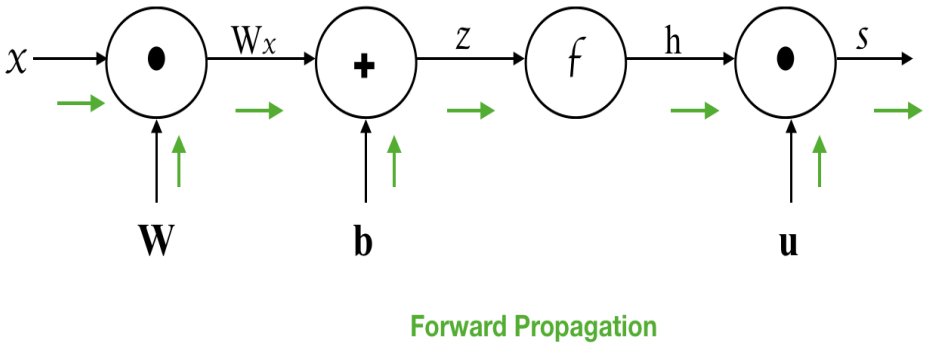

Figure 3: Forward Pass

In the forward pass the dot product of input to each neuron and weight is added with a bias. This resulting z is then activated using an activation function and passed on to the next layer as an input. Below equations are mathematical representation of the same.

$$input = x$$
$$z = Wx + b$$
$$h = f(z)$$
$$s = u^T h$$

For backward propagation we take the partial derivatives at each step and pass the gradient along edges in the backward direction. Below figure depicts the same [3]:-

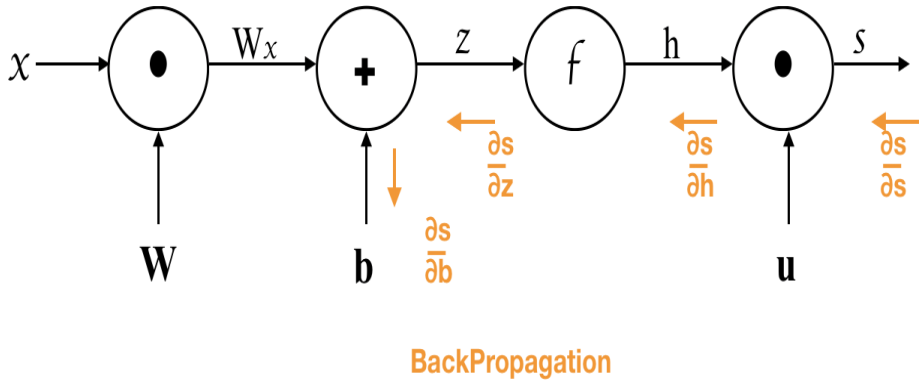

Figure 4: Backward Pass

Considering a single node, each node receives an upstream gradient and we accurately pass on the downstream gradient.

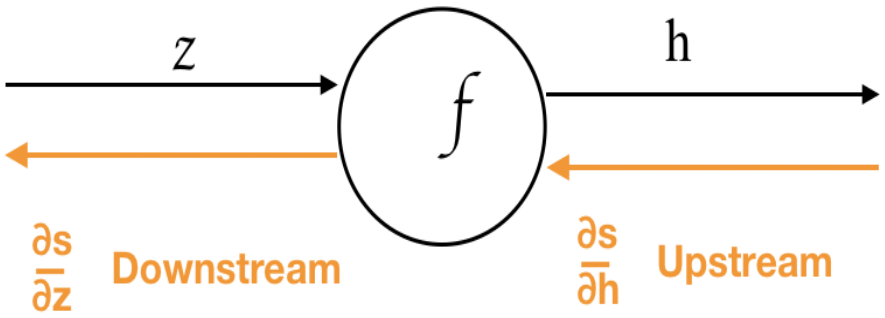

Figure 5: Single node gradients

Along with upstream and downstream gradient, each node has its own local gradient. Using the local gradient and upstream gradient we pass on the downstream gradient.

$$DownstreamGradient = UpstreamGradient X LocalGradient$$

From the figure 5,

$$LocalGradient = \frac{\partial h}{\partial z} \tag{1}$$

Using the chain rule,

$$\frac{\partial s}{\partial z} = \frac{\partial s}{\partial h}\frac{\partial h}{\partial z} \tag{2}$$

Using the chain rule we identify how much each weight affects overall error. The weights are then updated at each level using these derivatives.

5 Explain why we need structures to analyze language dependency.

Language is used by humans to communicate ideas, express sentiments. To understand or analyze a language we need a good knowledge of how the words in the language are dependent on each other. Basically the rules, grammar and structure of the language are important to correctly interpret the language. The structures reveal how the sentences is made up.

If the structure and grammar of the language is not used correctly then the meaning of the text might change or the sentence generated is meaningless. Example is as below:-

"Look at the dog with one eye."

The above sentence has two meanings either the dog has only one eye or use your only one eye to look at the dog.

For applications like machine translation, image caption generation, question and answer generation etc where the model is trying to generate a text in a particular language and without knowing the correct structure of that language the model will fail to produce good accurate texts as output.

6 Explain how n-gram technique is used for:

6.1 language modelling

Language modelling is defined as the process of predicting what word would come next in a given phrase. Suppose we have a vocabulary $V = w_1, w_2, w_3, ..., w_{|V|}$

A sentence is made up of the words in the given order $x_{(1)}, x_{(2)}, x_{(3)}...., x_{(t)}$

calculate the probability of the next word that comes after $x_{(t)}$ i.e. predicting $x_{(t+1)}$.

$$P(x_(t+1) \mid x_{(t)}, x_{(t-1)}...., x_{(1)})$$

N-gram as language models observes the frequency of different n-grams for a corpus and this data is then used to predict next word. n in n-grams is equal to the number of consecutive words taken into consideration. For one word (example- "I", "am", "about", "to", "board", "my") we use unigrams, similary for 2 words (example- "I am", "am about", "about to", "to board", "board my") is bigrams and so on. Basically, n-grams makes an assumption that the next word is dependent on previous n-1 words. To determine the probability of a n-grams based on n-1 gram:

$$P(x_(t+1) \mid x_{(t)}, x_{(t-1)}...., x_{(1)}) = \frac{P(x_{t+1}, x_t, ..., x_{t-n+2})}{P(x_t, ..., x_{t-n+2})} \tag{3}$$

we count the number of time n-1 grams and n grams appear in a corpus.

$$\approx \frac{Count(x_{t+1}, x_t, ..., x_{t-n+2})}{Count(x_t, ..., x_{t-n+2})} \tag{4}$$

Considering the above example "I am about to board my", to predict what comes next in this sentence using a 5-gram model:

$$P(\textbf{nextword} \mid about\ to\ board\ my) = \frac{Count(about\ to\ board\ my\ \textbf{nextword})}{Count(about\ to\ board\ my)} \tag{5}$$

The options of the next word could be train or flight.

$$P(train \mid about\ to\ board\ my) = \frac{Count(about\ to\ board\ my\ train)}{Count(about\ to\ board\ my)} \tag{6}$$

$$P(flight \mid about\ to\ board\ my) = \frac{Count(about\ to\ board\ my\ flight)}{Count(about\ to\ board\ my)} \tag{7}$$

Based on which n-gram has the highest probability given n-1 grams, in a given corpus that word is chosen out of the two. [4]

6.2 feature representation of text

N-grams as mentioned above, in a corpus takes into account the frequency of a group of words occurring together. This helps us identify dependencies among words of a language. Using the variants of n-grams like unigrams, bigrams, trigrams etc we can extract the feature representation of a particular text. The main goal is to represent sentences in a corpus into a vector that can be used as input/output for a model.

Consider a corpus consisting of 10 sentences

- Extract sentences from the corpus one by one.

- Converting the entire corpus to lower case and removing the punctuation is pre-processing done usually for a corpus.
- Use these lines to form a vocabulary of unique words, the vocabulary basically depends on the n-gram being used. For unigrams it is all the unique words in the corpus, for bigrams it is combination of two words in each line and so on for higher grams.
- Using n-grams these sentences in the corpus can be represented as features. n-grams will create feature representation for each sentence, which has as many elements as the number of unique words(length of the vocabulary) in the corpus. Suppose for uni-grams the vocabulary size is 10 then each sentence will be represented by a feature of size 10. It counts the frequency of each unique word in a particular sentence. If a word does not appear in sentence then it is marked as 0.
- Similarly, bigrams create a vocabulary using two words together and then represent the same sentences as features that contain the frequency of group of 2 words. The same applies to higher grams.

Feature representation of text extracted from n-grams hold information about the dependency of words i.e. which word appears more frequently with another word or a sequence of words.

7 n-grams and Binary Classification Implementation

7.1 Project Structure

The implementation is in the file **4736715_Jaisinghani_hw2sol.py**, need to run this. It consists of following methods:-

- **ngrammodelclassification** - This method contains the core code for n-gram feature representation, 10-fold cross validation, binary classification.
- **readFile** - This method is used to read the input file line by line and stores it in a variable.
- **dataPrep** - This method handles the punctuation in the corpus. To get the correct unique words or vocabulary from the dataset, I have replaced the characters ,.!?+=*-():;[]/ with a space and removed rest other punctuation.
- **dataPreProcessing** - This method is used to separate text from their labels and also make separate dataset for 0 labels and 1 labels. It returns three dataframes of pandas. The dataFrame consists of all the sentences and their labels, dataFrame_0 consists of all the texts and labels for the label 0 and dataFrame_1 consists of all the texts and labels for the label 1.
- **window** - This method takes in the sentence and using the start index and end index returns words by sliding the window over the sentence.
- **createVocab** - This method returns the vocabulary for a specified n i.e. for n = 1 returns the vocabulary for f1, n=2 returns vocabulary for f2 and so on.
- **getColumnNumber** - Fetches the column number to be updated in an array.
- **createngrams** - This methods create the feature vector representation for unigrams. It takes the dataframe, the vocabulary for unigram and returns a 2d array with feature vector representation of unigrams.
- **generatengrams** - This methods create the feature vector representation for n=2 to 5. It takes the dataframe, the vocabulary for the respective n and returns a 2d array with feature vector representation of it.
- **getdataframedata, gettestdataframedata, getFeaturesData** - These methods are used to extract the required data from the dataframe for training and testing.
- **evaluate_prediction** - This method takes in the predicted value of the classifier and the actual labels and returns the accuracy.
- **confusionmatrix** - This method creates a confusion matrix based on the predictions and actual labels and returns true_positive, false_positive, true_negative, false_negative.

- **calculateMetrix** - This method calculates the metrics precision, recall, true positive rate, false positive rate, true negative rate, false negative rate.

7.2 Dataset

Validated my implementation on three datasets- amazon_cells_labelled.txt, imdb_labelled.txt, yelp_labelled.txt. Each file consists of about 1000 reviews from amazon, imdb and yelp respectively. Each of the review is labeled with 0 for bad review and 1 for a good review.

7.3 Steps for n-gram features and binary classification implementation

- The method ngrammodelclassification is called for every dataset i.e. the input file. Read the input file line by line and pre-processed it by removing punctuation and converting all the words to lower case.
- Used pandas dataframe to store the sentences, labels and their respective features. Created two different dataframes for labels 0 and 1.
- Made a list of vocabulary for the value of n= 1 , i.e. the vocabulary for f1.
- Using the vocabulary created, created a feature representation f1 for all the sentences in dataframe_0 and dataframe_1 by counting the number of times a particular word from the vocabulary appears in each sentence. Stored the feature under the column **text_f1** in the dataframe_0 and dataframe_1 based on the labels of the sentence.
- Followed the above to steps for f2, f3, f4, f5 to get the feature representations **text_f1, text_f2, text_f3, text_f4, text_f5** respectively and stored them under dataframe_0 and dataframe_1.
- To get the feature representations for combined features f1f2, f1f2f3, f1f2f3f4, f1f2f3f4f5 concatenated the respective feature representations.
- Using np.array.split, divided both the dataframes dataframe_0 and dataframe_1 into 10 parts for 10-fold cross-validation.
- For every feature f1, f2, f3, f4, f5, f1f2, f1f2f3, f1f2f3f4, f1f2f3f4f5 performed 10-fold cross validation by picking one part from each dataframe_0 and dataframe_1 as test set and the remaining 9 parts of each dataframe_0 and dataframe_1 as training set. Used the binary classifier logistic regression for the same.
- Calculated the false positive rate, false negative rate, true positive rate, true negative rate, accuracy, precision and recall for all the features and for every dataset.

7.4 Results

Below are the results for each dataset:

7.4.1 amazon_cells_labelled.txt

The below table depicts the false positive rate, false negative rate, true positive rate, true negative rate, accuracy (in percentage), precision and recall that was observed for amazon dataset:

Features	False Positive Rate	False Negative Rate	True Positive Rate	True Negative Rate	Accuracy	Precision	Recall
f1	0.14800	0.20800	0.79200	0.85200	82.20	0.84368	0.79200
f2	0.24800	0.27999	0.72000	0.75200	73.59	0.74886	0.72000
f3	0.6400	0.09000	0.90999	0.3600	63.50	0.58753	0.90999
f4	0.91599	0.01800	0.98200	0.08399	53.30	0.51749	0.98200
f5	0.98000	0.00400	0.99600	0.0200	50.80	0.50407	0.99600
f1f2	0.14000	0.19400	0.80600	0.86000	83.29	0.85344	0.80600
f1f2f3	0.14400	0.20800	0.79200	0.85600	82.40	0.84828	0.79200
f1f2f3f4	0.17800	0.16799	0.83200	0.82200	82.69	0.82564	0.83200
f1f2f3f4f5	0.15200	0.22000	0.78000	0.84800	81.40	0.83902	0.78000

Figure 6: Results for Amazon dataset

7.4.2 imdb_labelled.txt

The below table depicts the false positive rate, false negative rate, true positive rate, true negative rate, accuracy (in percentage), precision and recall that was observed for imdb dataset:

Features	False Positive Rate	False Negative Rate	True Positive Rate	True Negative Rate	Accuracy	Precision	Recall
f1	0.23200	0.20200	0.79799	0.76800	78.30	0.77869	0.79799
f2	0.29800	0.41200	0.58800	0.70200	64.50	0.66575	0.58800
f3	0.16400	0.65400	0.34600	0.83600	59.10	0.68120	0.34600
f4	0.03000	0.89399	0.10600	0.97000	53.80	0.78192	0.10600
f5	0.00800	0.96600	0.03400	0.99200	51.20	0.75000	0.03400
f1f2	0.25000	0.23600	0.76400	0.75000	75.70	0.75473	0.76400
f1f2f3	0.25200	0.23400	0.76599	0.74800	75.70	0.75306	0.76599
f1f2f3f4	0.32399	0.18200	0.81800	0.67600	74.70	0.71813	0.81800
f1f2f3f4f5	0.27200	0.24200	0.75800	0.72800	74.30	0.73566	0.75800

Figure 7: Results for imdb dataset

7.4.3 yelp_labelled.txt

The below table depicts the false positive rate, false negative rate, true positive rate, true negative rate, accuracy (in percentage), precision and recall that was observed for yelp dataset:

Features	False Positive Rate	False Negative Rate	True Positive Rate	True Negative Rate	Accuracy	Precision	Recall
f1	0.18599	0.19200	0.80800	0.81400	81.10	0.81445	0.80800
f2	0.30800	0.19200	0.80800	0.69200	75.00	0.72489	0.80800
f3	0.66000	0.11000	0.88999	0.33999	61.50	0.57484	0.88999
f4	0.89000	0.02400	0.97600	0.11000	54.30	0.52308	0.97600
f5	0.97399	0.00200	0.99800	0.02600	51.20	0.506131	0.99800
f1f2	0.18000	0.19000	0.81000	0.82000	81.50	0.81998	0.81000
f1f2f3	0.18800	0.19000	0.81000	0.81199	81.10	0.81323	0.81000
f1f2f3f4	0.22799	0.16000	0.84000	0.77200	80.60	0.78884	0.84000
f1f2f3f4f5	0.19400	0.19400	0.80600	0.80600	80.60	0.80972	0.80600

Figure 8: Results for yelp dataset

References

[1] *https://www.analyticsvidhya.com/blog/2014/10/ann-work-simplified/.*

[2] *Meet Artificial Neural Networks https://towardsdatascience.com/meet-artificial-neural-networks-ae5939b1dd3a.*

[3] *Natural Language processing - Lecture 5 by Dr. Aziz Mohaisen.*

[4] *Natural Language processing - Lecture 7 by Dr. Aziz Mohaisen.*