International Institute of Information Technology
Hyderabad

Language Technologies Research Centre

Project Report

Question Type Classification

June 27, 2021

Contributors

- siva prasad reddy garlapati
- phanisukesh
- dharani priya
- priyanka betha

Contents

1 Introduction **3**
1.1 Aims and Objectives 3

2 Related Work **4**
2.1 Literature Review 4
2.2 Approach 4

3 Corpus Description **5**
3.1 Corpus 5
3.2 Statistics 5
3.3 Feature Engineering 6
3.3.1 Count Vectors as features : 6
3.3.2 TF-IDF Vectors as features : 7
3.3.3 Word Embeddings as features : 7

4 Model Implementation **7**
4.1 Logistic Regression 8
4.2 Naive Bayes 9
4.3 Random Forest 10
4.4 Support Vector Machine 12
4.5 Multi-layer perceptron 13
4.6 Convolutional Neural Network 15
4.7 Recurrent Neural Network 16
4.7.1 Long-Short Term Memory 16
4.7.2 Gated Recurrent Units 17
4.8 Bidirectional Encoder Representations from Transformers 18

5 Experiments And Results **18**
5.1 Experiments 18
5.2 Results 19

6 Future Work **19**

7 Conclusions **20**

1 Introduction

Languages play a vital role in communication whether it is among humans or machines. Around 6500 languages are spoken aroud the world, among them 1652 are from India. In the modern era of communication, everyone wants to get all the information in their native language. Telugu is one of the widely spoken languages in southern India. But, except for some dialogue based Question Answering (QA) systems, no other model is implemented to do QA in web-based applications for the Telugu language. So we choosed the Telugu QA task with a pre-processed data set of 1340 Queries. QA system is built with two techniques (Information retrieval and extraction).

The Question Type classification (QC) for Telugu is an essential part in identifying the NER type of question in the Open-domain Question Answering system (QA) for a low resource language like 'Telugu'. One of the prime modules of QA system is Question Classifier (QC) which is not available for Telugu. Also there is no pre-tagged data (query categories)available for Telugu, which makes Telugu QC is a challenging task. It indirectly affecting the accuracy of QA system. It is a necessary module for to correctly answer user's questions in QA system, the system has to know what the users are looking for, and it is QC that presents important searching clues for the system. Here we took PERSON, NUMBER, LOCATION, ORGANISATION, DATE, TIME, MONEY, PERCENTAGE as our answer type classes. To design a Telugu QA model we used a data set for QC module and then we described Question classification through machine learning approaches, namely, with Statistical and Deep Neural Network models. In statistical models,the highest accuracy obtained, with TFIDF character level vectorization for SVM classifier as 88 percentage. And in Deep Neural Network Techniques using FasText Telugu Embeddings 90 percentage and with Byte-Pair-Encoding Telugu Embeddings 86 percentage for Bidirectional RNN model. Finally with BERT model we got an accuracy as 91 percentage.

1.1 Aims and Objectives

This project aims to obtain the answer type of a given Telugu question or query by identifying Named Entity Recognition (NER) type of a given question.

We applied statistical models like LR, MLP, SVM, NB, RF classifier models and Deep Neural Networks models like CNN, RNN-LSTM, RNN-GRU, Bidirectional RNN models. And also we also applied model namely BERT in which we can increase our model accuracy with small Dataset using pre-trained models available in transformers library. All classifiers need input to be in the form of vector representation. So we used TFIDF and Count Vectorisation techniques for questions and used as features for statistical classification models. And for Deep Neural Network models we took Telugu Embeddings to represent our questions as vectors.

2 Related Work

2.1 Literature Review

We have gone through many research papers for Question Type Classification (QC). We came to know that many are implimented for English language only. taking them as reference we found few steps and Approach to implement our project QC.

Dataset Preparation : The first step is the Dataset Preparation. It includes the process of loading a dataset and performing basic pre-processing. The dataset is then splitted into train and validation sets.

Feature Engineering : The next step is the Feature Engineering in which the raw dataset is transformed into flat features which can be used in a machine learning model. This step also includes the process of creating new features from the existing data.

Model Training : The final step is the Model Building step in which a machine learning model is trained on a labelled dataset.

2.2 Approach

We found a paper [2] QC using SVM for English language. With that we got some idea. Then we found other research paper [4] named Open-Domain Question Answering system for Telugu language. In this they used a QC module for correct prediction of type of answers the question results. In this they used Logistic Regression (LR), Multilayer Perceptron (MLP), Support Vector Machine (SVM) classifiers with Term Frequency Inverse Document Frequency (TFIDF) word level vectorization. That means each query is considered as a document and each word as a term. In MLP classifier they considered one input layer, four hidden layers and one output layer, with 'stochastic gradient descent' as a loss optimizer and tanh activation. In case of LR classifier used multiclass as 'multi-nominal' with 'stochastic average gradient' optimizer. Another classifier SVM, taken multi-class as 'one-vs-rest', with loss as 'squared-hinge'. They got accuracies in percentage as 71 for MLP, 72 for LR, 73 for SVM.

To increase the accuracy we have implemented those models with Count vectors and TFIDF n gram and character level vectorisation. In addition to those we have implemented using Deep Neural Network models using pre-trained FastText and Byte-pair-Encoding Embeddings based on this reference [1]. And also we implemented with BERT based on found in [3] which we use already trained models which we get from Transformers library.

3 Corpus Description

3.1 Corpus

NLP projects are basically built using a large amount of data. "Corpus" is a collection of structured written or spoken natural language material, stored on computer, and used as input data to find out the facts that can help us to develop NLP applications.

This QC is learnt in supervised manner. So each Telugu question is labeled with its Answer type. Telugu questions are prepared on different categories which answer types are like numbers, location, person, date, money, time, percentage, organisation etc.,. Each question/query is representing as vectors and then used for Training the classifier. The labels are converted into some unique integers. The structure or data format of our corpus to feed as input for this Question Type Classification is as below

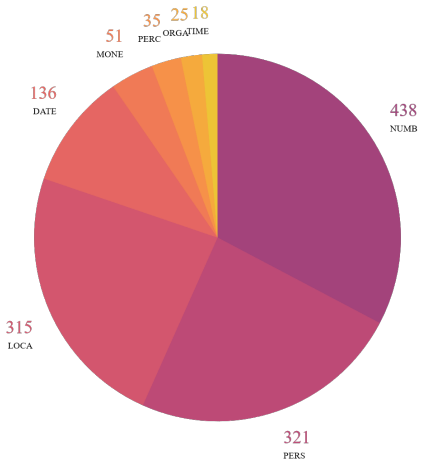
(label)(colon(:))(TAB)(question)

3.2 Statistics

After performing some data analysis on this structured corpus data. We have total of 1339 questions/queries with their answer type. We have 8 categories of answer types namely Number, Person, Location, Date, Money, Time, Percentage, Organisation. Each label counts are as shown in figure

Category	Label	Number of questions
Number	NUMB	438
Person	PERS	321
Location	LOCA	315
Date	DATE	136
Money	MONE	51
Time	TIME	35
Percentage	PERC	25
Organisation	ORGA	18

Table 1: Labels counts



3.3 Feature Engineering

In this step, raw data of questions/queries which is in text be transformed into feature vectors which means set of numbers. We will implement the following different ideas in order to obtain relevant features by removing all non useful data like punctuations, special characters, from our dataset. Those are

- Count Vectors as features
- TF-IDF Vectors as features
- Word Embeddings as features

Before going to see about those we need to know some definitions as term, document of Corpus. We know that from previous sections corpus is typically a collection of text documents. So all the sentences in the dataset are treated as document and terms may be vocabulary of words or n-grams of words or characters of words.

3.3.1 Count Vectors as features :

Count Vector is a matrix notation of the texts or sentences of dataset in which every row represents a document from the corpus, every column represents a term from the corpus, and every cell represents the frequency count of a particular term in a particular document.

3.3.2 TF-IDF Vectors as features :

In this we calculate Term Frequency-Inverse Document Frequency (TF-IDF) score. It represents the relative importance of a term in the document and the entire corpus. TF-IDF score is calculated by two terms. We represent that score as **tfidf(w, D)** which is product of tf(w, D), idf(w, D) for word **w** in document **D**. The term **tf(w, D)** represents the term frequency of the word w in document D. The term **idf(w, D)** is the inverse document frequency for the term w, which can be computed as the log transform of the total number of documents in the corpus C divided by the document frequency of the word w, which is basically the frequency of documents in the corpus where the word w occurs.

In this we have 3 types based on the term we choose. If we take words as our terms then it is word level TF-IDF vectorization. If we take terms as n-grams of the words of the documents then it is n-gram level TF-IDF vectorization. If we take characters of words of the document as our terms then it is character level TF-IDF vectorization.

3.3.3 Word Embeddings as features :

A word embedding is a form of representing words and documents using a dense vector representation. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. Word embeddings can be trained using the input corpus itself or can be generated using pre-trained word embeddings such as Glove, FastText, Byte-Pair Encoding and Word2Vec. Any one of them can be downloaded and used as transfer learning that is already existing embedding which suits for our task is used to create features for our document.

So in this we used pre-trained 300 dimension Telugu FastText and Byte-Pair Encoding word embeddings. To make model using these, we have 4 steps to implement. Those are Loading the pretrained word embeddings, Creating a tokenizer object, Transforming text documents to sequence of tokens and pad them, Create a mapping of token and their respective embeddings.

In these ways we can represent out text document that is our questions with numbers then used in classifier learning and to create models.

4 Model Implementation

For our problem statement we are using Machine learning Algorithms. Specifically ours is Text classification problem which is under Natural Language Processing. The goal of Text classification is to automatically classify the text documents based on categories

we have. Text Classification is an example of supervised machine learning task since a labelled dataset containing text documents and their labels is used to train a classifier. In this we have

Binary or binomial classification is exactly for two classes to choose between (usually 0 and 1, true and false, or positive and negative)

Multiclass or multinomial classification is for three or more classes of the outputs to choose.

So ours is Multiclass classification problem. In this text documents are Telugu questions and categories are Number, Person, Location, Date, Money, Time, Percentage, Organization.

The final step in the text classification framework is to train a classifier using the features created in the previous step. There are many different choices of machine learning models which can be used to train a final model. We will implement following different classifiers for our task of retreiving label of answer type of given Telugu question are as below

- Logistic Regression
- Naive Bayes Classifier
- Random Forest Model
- Support Vector Machine
- Multi-layer Perceptron
- Deep Neural Networks

4.1 Logistic Regression

Logistic regression is a very popular machine learning technique. It belongs to the group of linear classifiers. We use logistic regression when the dependent variable is categorical. It is somewhat similar to polynomial and linear regression. The implementation of Multiclass classification follows the same ideas as the binary classification. As we know in binary classification we solve for two categories. In that case we assign 0 for one category and 1 for other. But when working with multi class we follow an approach of one vs rest method. In the one vs rest method, when we work with a class, that class is denoted by 1 and the rest of the classes becomes 0.

We could start by assuming p(x) be the linear function where x denotes our features of our text. However, the problem is that p is the probability that should vary from

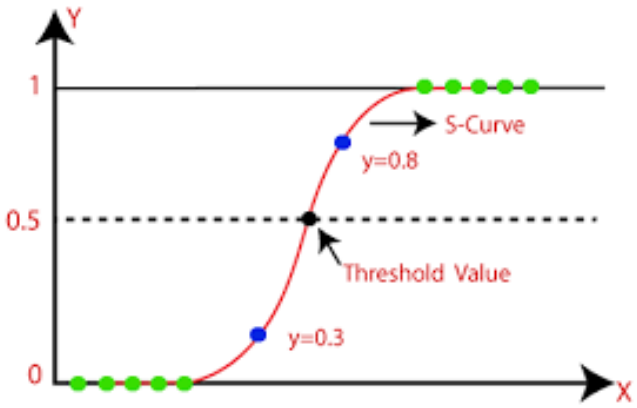
0 to 1 whereas p(x) is an unbounded linear equation. To address this problem, let us assume, log p(x) be a linear function of x and further, to bound it between a range of (0,1), we will use logit transformation using sigmoid function.It squeeze the values from (-k,k) to (0,1). Therefore, we will consider log p(x)/(1-p(x)). Next, we will make this function to be linear as below

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \alpha_0 + \alpha.x$$

after solving it

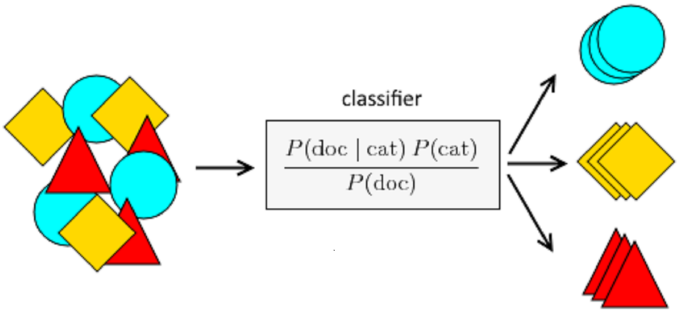
$$p(x) = \frac{e^{(\alpha_0+\alpha)}}{e^{(\alpha_0+\alpha)}+1}$$

Then we take a threshold such as 0.5. If p(x) function returns a value greater than or equal to 0.5, we take it as 1, and if the sigmoid function returns a value less than 0.5, we take it as 0.



4.2 Naive Bayes

Naive Bayes is a simple technique for constructing classifiers models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.



For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. Bayes theorem provides a way of calculating posterior probability as below

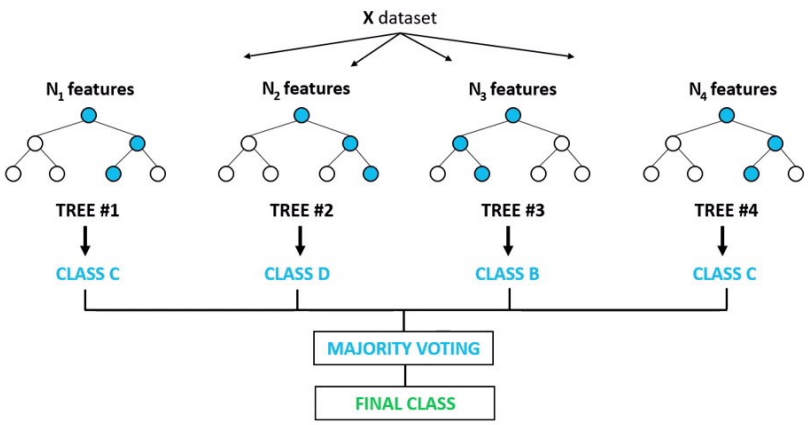
$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Above,

P(c given x) is the posterior probability of class (c is label) given predictor (x is attributes).
P(c) is the prior probability of class.
P(x given c) is the likelihood which is the probability of predictor given class.
P(x) is the prior probability of predictor.

4.3 Random Forest

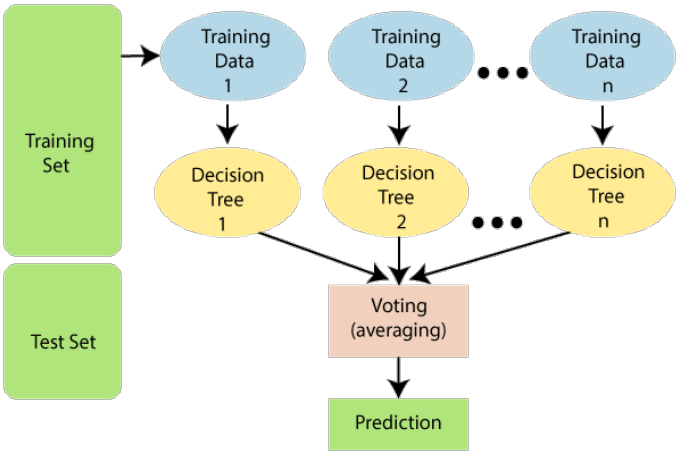
Random Forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.



As in example of above fig, A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Random forest algorithm works in four steps:

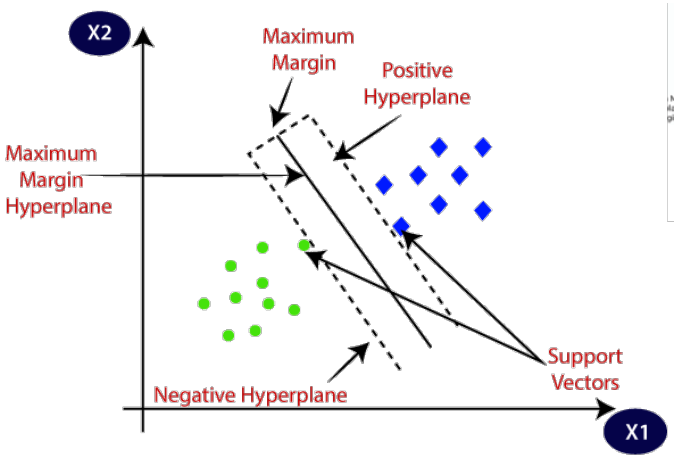
1. Select random questions from a given dataset.
2. Construct a decision tree for each question and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.



4.4 Support Vector Machine

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple. The algorithm creates a line or a hyperplane which separates the data into classes. SVM is an algorithm that takes the data as an input and outputs a line that separates those classes.

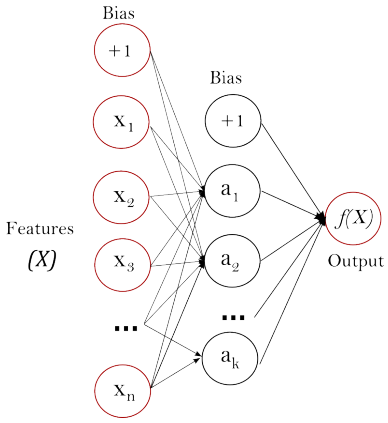
According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane. If data is not linearly separable, we cannot draw a straight line that can classify the data. But, this data can be converted to linearly separable data in higher dimension. Lets add one more dimension with existing dimensions, such that it can linearly separable with hyperplane and then projecting the decision boundary back to original dimensions using mathematical transformation. We can use kernels which automatically does all these steps in sklearn's SVM to implement.



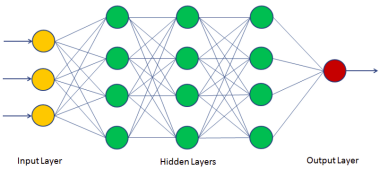
In its most simple type SVM are applied on binary classification, dividing data points either in 1 or 0. For multiclass classification, the same principle is utilized. The multiclass problem is broken down to multiple binary classification cases. Like in Logistic Regression it also follows the approach of one vs rest. So it divides the data points in class x and rest. Consecutively a certain class is distinguished from all other classes.

4.5 Multi-layer perceptron

Multi-layer perceptron (MLP) is a supervised learning algorithm that learns a function f(.):Rm to Ro by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X = x1,x2,....xm and a target y, it can learn a non-linear function approximates for either classification and regression. In this we have input and the output layer, there can be one or more non-linear layers, called hidden layers. As in below figure



The circles represents neurons with two mathematical operations. Every input is given with random weights, so in neuron, this weighted input sum is calculated and normalises that sum value using some activation function. Again those outputs acts as input to next layer. More the hidden layers that that much nice our model works for our problem statement.



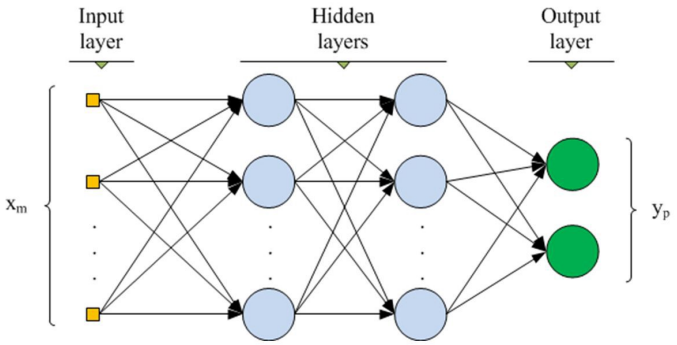
So for our case also it works in the same way but only change in the output layer. Since we are having 8 categories for our QC we use 8 target values. This output layer results 8 output probabilities, in that the highest probable category is chosen for given query or question.

These are the statistical models in machine learning. So, upto the discussed classifier till now uses Count and TFIDF vectors, and they acts as input feature vectors to train a classifier.

Deep Neural Networks :

A Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions.[112] These components functioning similar to the human brains and can be trained like any other ML algorithm as above we discussed.

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1 through activation function in neuron. If the network did not accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.



In this techniques we implement Convolutional Neural Network (CNN) model, Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) model, Gated recurrent units Recurrent Neural Network (GRU-RNN) model, Bilateral Recurrent Neural Network (Bi-RNN) models. They all seems to be very similar only change in one layer.

In these models we use Word Embeddings as features. We already discussed that in this the questions are transformed to padded sequence based on the indexes of words created by tokenizer. We give input as this sequence of questions to create DNN models. We have created 5 types of layers in creating model. They are Input layer which accepts the sequence of fixed input sequence shape, then that sequence of indexes are converted converted into matrix of 300 dimension using Embedding matrix created by pre-trained Embeddings with regularisation like Spatial Dropout, then it is given to specific model layer like CNN, RNN layers, then it is given to fully connected DNN layer with one Dense layer of 128 neurons with regularization parameter, Dropout as

0.25. Then this outputs are connected to last output layer of 8 neurons each corresponds one category with softmax as activation that it make sure the resulted values of this layer sums to 1. These are are probabilities of appearing that particular category. Based on the maximum probability of those 8 neurons outputs is chosen as category for given sequence of question.

4.6 Convolutional Neural Network

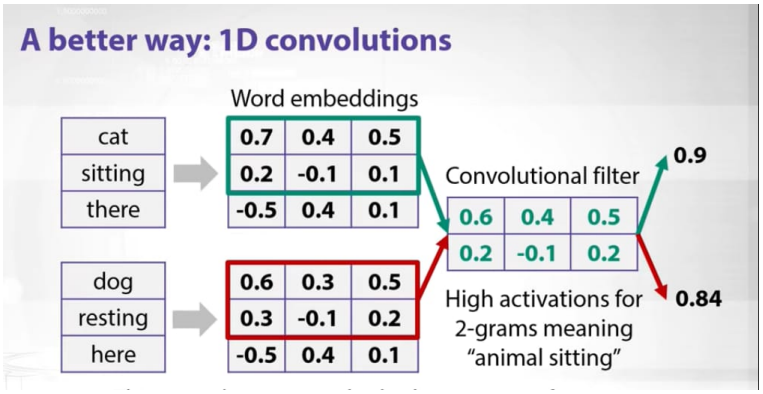
Convolutional Neural Networks, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.

The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks. In a regular Neural Network there are three types of layers:

Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data.

Hidden Layer: The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layers can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.

Output Layer : The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.

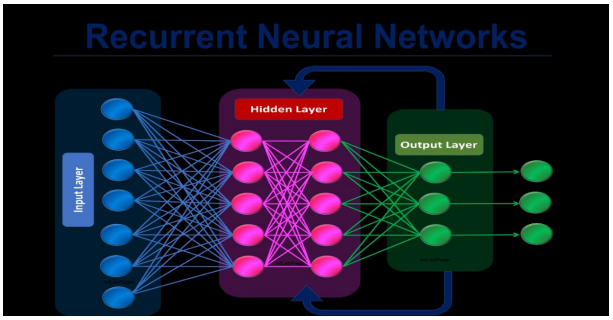


So in this we used Convolution in one direction. We have filters in this, which slides on our embedding matrix formed in Embedding layer. We have taken 256 filters of size 3 means it gives the convoluted sum of 3 rows of embedding matrix with filter each time in up to down direction as in above figure. But in this every word is represented with 300 dimension vector using the above mentioned pre-trained embeddings. Then every convolutional layer followed by pooling layer in which we can remove the noise, unwanted features and reduces the size of matrix.

4.7 Recurrent Neural Network

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

RNN have a "memory" which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

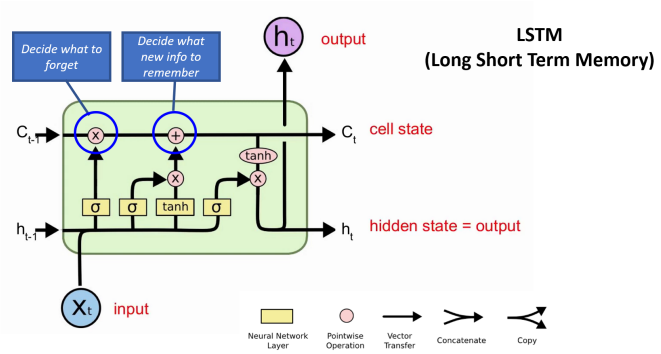


4.7.1 Long-Short Term Memory

LSTMs are a special kind of Recurrent Neural Network — capable of learning long-term dependencies by remembering information for long periods is the default behavior. All recurrent neural networks are in the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have a chain-like structure, but the repeating module is a bit different structure. Instead of having a single neural network layer,

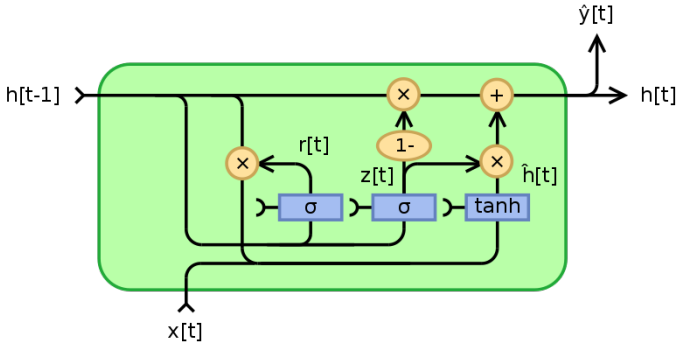
four interacting layers are communicating extraordinarily.

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.



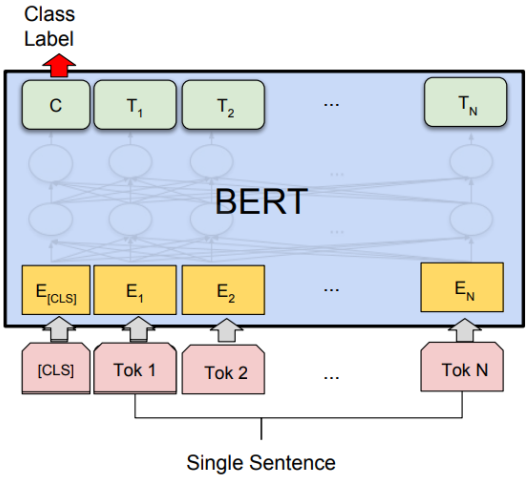
4.7.2 Gated Recurrent Units

GRU supports gating and a hidden state to control the flow of information. To solve the problem that comes up in RNN, GRU uses two gates: the update gate and the reset gate.The update gate is responsible for determining the amount of previous information (prior time steps) that needs to be passed along the next state. It is an important unit. The below schema shows the arrangement of the update gate.the reset gate is used from the model to decide how much of the past information is needed to neglect. The formula is the same as the update gate. There is a difference in their weights and gate usage, which is discussed in the following section. The below schema represents the reset gate.



4.8 Bidirectional Encoder Representations from Transformers

Transformer models have been showing incredible results in most of the tasks in natural language processing field. The power of transfer learning combined with large-scale transformer language models has become a standard in state-of-the art NLP.



It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

This pre-training step is half the magic behind BERT's success. This is because as we train a model on a large text corpus, our model starts to pick up the deeper and intimate understandings of how the language works. This knowledge is the swiss army knife that is useful for almost any NLP task.

In this task of labelling a answer type of Telugu question we use a pre-trained model name telugu-bertu-tydiqa.

5 Experiments And Results

5.1 Experiments

As we have seen in 2.1, the steps to proceed for this Text classification are Dataset Preparation, Feature Engineering and then Model Training.

For this, we are using dataset of Telugu questions which is prepared on different categories of answer types. The dataset consists of 1339 text questions and their labels. To prepare the dataset, load the data into a pandas Dataframe by adding the questions and labels in a lists. Extract features using Feature Engineering techniques as mentioned in 3.3. Then encode target column with unique numbers so that it can

be used in machine learning models. And then split the dataset into training and validation sets such that every classifier in section 4, we tarin using training set and test on validation set. To know the performance of each classifier we can calculate accyracy score that how the classifier works on unseen or untrained data.

5.2 Results

We implemented the classifiers mentioned in section 4 on our training set. We calculated accuracy on the unseen validation set. It is reported in the below table. We included the accuracy obtained with Statistical Models by taking TF-IDF word level, n-gram level, char level vectors as feature vectors and in addition to that we used Count vectors as also feature vectors. Deep Learning models accuracies are also evaluated on same validation set with FastText and Byte-Pair-Encoding pre-trained Telugu word embeddings as feature vectors. And with BERT using telugu-bertu-tydiqa as pre-trained model, we got an accuracy as 91 percentage which is highest of all the models.

		WordLevel	N-gram	CharLevel	Count level vector
Statistical Method	Logistic Regression	81	71	82	78
	Support Vector Machine	80	72	83	74
	Multi Layer Perceptron	80	71	86	74
	Naive Bayes	73	73	77	76
	Random Forest	80	76	84	77
	decision tree	71	71	75	71

Table 2: Accuracies with Statistical models

6 Future Work

In this paper we predicted the output by using different methods discussed in the above. The result and the accuracies are good and future work can be focused on ob-

		Fast Text	Byte-Pair
Deep Learning	Convolutional Neural Network	84	82
	Recurrent Neural Network-LSTM	83	83
	Recurrent Neural Network- BI-LSTM	84	84
	Recurrent Neural Network-GRU	84	84

Table 3: Accuracies with Deep Learning

Method	Accuracy
Bidirectional Encoder Representations from Transformers	85

Table 4: Accuracy with BERT

taining more accuracy. In future we want to implement this by using XLM - Roberta pre-trained model to obtain the better accuracy.

XLM - Roberta is a multilingual model trained on 100 different languages. Unlike some XLM multilingual models, it does not require tensors to understand which language is used, and should be able to determine the correct language from the input ids.

7 Conclusions

This Question type classification aim is to predict the correct label for the given input query. Firstly we have trained our dataset and predicted the output by using the statistical models like LR, SVM, MLP, RF, NB with different features like word-level, n-gram vectors, char-level vectors, count vectors to convert the data into vectors. In this we obtained more accuracy and correct label with char-level vectors. To get more accuracy we have used deep learning Techniques using FastText Pre trained Embedding and Byte-Pair Encoding Pre trained Embedding. We got more accuracy by using FastText Pre trained Embedding with correct label. And finally we used BERT(Bidirectional Encoder Representations from Transformers) and this is most accurate than the above used methods and predicted the correct label for the given query.

References

[1] Shivam Bansal. A comprehensive guide to understand and implement text classification in python. *Analytics Vidhya*, 2018.

[2] Kadri Hacioglu and Wayne Ward. Question classification with support vector ma-

chines and error correcting codes. In *Companion Volume of the Proceedings of HLT-NAACL 2003-Short Papers*, pages 28–30, 2003.

[3] Sunsan Li. Text classification with bert. `https://github.com/susanli2016/NLP-with-Python/blob/master/Text-Classification-With-BERT`, 2016.

[4] Priyanka Ravva, Ashok Urlana, and Manish Shrivastava. Avadhan: System for open-domain telugu question answering. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pages 234–238. 2020.