School of Engineering and Applied Science (SEAS), Ahmedabad University

B.Tech (ICT) Semester VI / M.tech / PhD: Machine Learning (CSE 523)

- Group No : B_NLP3
- Group Members:
  1. Yesha Shastri (AU1741035)
  2. Ratnam Parikh (AU1741036)
  3. Devshree Patel (AU1741075)
  4. Param Raval (AU1741083)
- Project Title: Text Generation and Sentiment Analysis using BERT
- Project Area: Natural Language Processing

1 Introduction

1.1 Background

- Question-Answering is an important application of NLP in real life though it does not come without its challenges. One approach to get a machine to answer questions is Reading Comprehension (RC). Reading a text and answering from it is a challenging task for machines, requiring both understanding of natural language and knowledge about the world. The first step in the process to create such a system is a Question Answering dataset. A popular benchmark QA dataset created by Stanford University is known as the Stanford Question Answering Dataset or SQuAD. The detailed description of the dataset is given in next section of the report. The second part is on Text Generation or answering in our case. It is sub-field in natural language processing, that provides knowledge in computational linguistics and artificial intelligence to generate text that can contain certain context and therefore give a meaning to the text generated.

- There are several question-answering datasets created to date to address different problems in NLP including Stanford Question-Answering Dataset (SQuAD), Natural Questions (NQ), Question Answering in Context (QuAC), Conversational Question Answering (Coca), HOTPOTQA, ELI5, ShARC, MS MARCO, TWEETQA, NEWSQA. From these datasets, SQuAD is a well-defined and well-structured dataset and related work for background knowledge is also available. Moreover, it approximates the real reading comprehension circumstances and models that have great performance on SQuAD can be regarded as a solid benchmark to solve RC and QA problems (BERT for SQuAD 2.0).

- One of the first papers on applying machine learning approach for question answering tasks is Ng. et al., 2000. A semi-supervised learning approach for word representations was given by Turian et al., 2010. A fast hierarchical language model was proposed by Mnih et al., 2009 which outperforms non-hierarchial neural models and best N-gram models. The task of QA is always challenging since it requires a comprehensive understanding of natural languages and the ability to do further inference and reasoning. In the original June, 2016 paper introducing SQuAD, Rajpurkar, et al [1]. developed a logistic regression model based on detailed linguistic features that achieved an F1 score of 51%. Then, Xiong, et al. introduced a dynamic coattention network for encoding and attention modeling, Seo, et al. utilize bidirectional attention flow (BiDAF) encoding and attention schemes, combining question and context influence via a bidirectional LSTM. Later, using the self-attention based Transformer by Vaswani et. al.[2][4], Devlin, Jacob et. al. [3] from Google AI release the Bidirectional Encoder Representations and Transformer (BERT) model which surpassed the performance of all previous SOTA models on SQuAD 1.1.

1.2 Motivation

While Google already implements BERT and more advanced models to process text, question-answering and text generation can be applied to e-commerce platforms for generating factual answers to user queries and comments. Similarly, since the model can be trained well on context specific data, an effective application would be a chatbot for the healthcare sector. The chatbot can be trained on existing medical corpus data (such as mental health) and can adjust based on what it learns about its user. Since, there are already several advances and modifications made to BERT in literature, applications in several such domains remain unexplored in the mainstream.

1.3 Problem Statement

- While processing text data for question-answering, it is imperative that the machine understands the contextual sense of the sentences and processes them almost like a human to answer correctly. Thus, our main problem is to generate sentence representations and context-informed word embeddings with weights that closely resemble the significance and contextual meaning of the sentence.

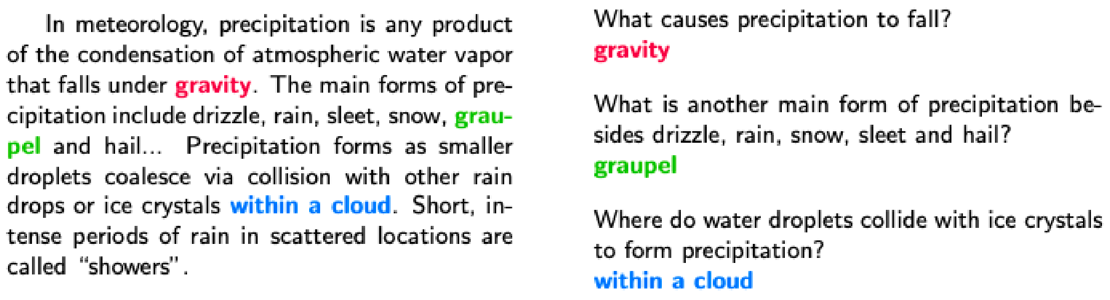
  Here, we compare results of various popular Machine Learning models on one particular sentence encoding method with the more complex BERT model which employs Deep Learning methods like DNN. Since the aim is to find the answer that fits closest to what we have in the document, we use cosine similarity between the target sentence and the question as one of the features. Otherwise, the models are compared using the standard test accuracy metric of measure.

2 Data Acquisition / Explanation of Data set

- **Explanation of Dataset**

  We have used the Stanford Question Answering Dataset(SQuAD), which is a reading comprehension dataset consisting of nearly 100,000 questions posed on several Wikipedia articles. The answer to each question is a segment of text from the corresponding reading passage.

  Sample passage from the dataset with question-answer pairs is shown below.



In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

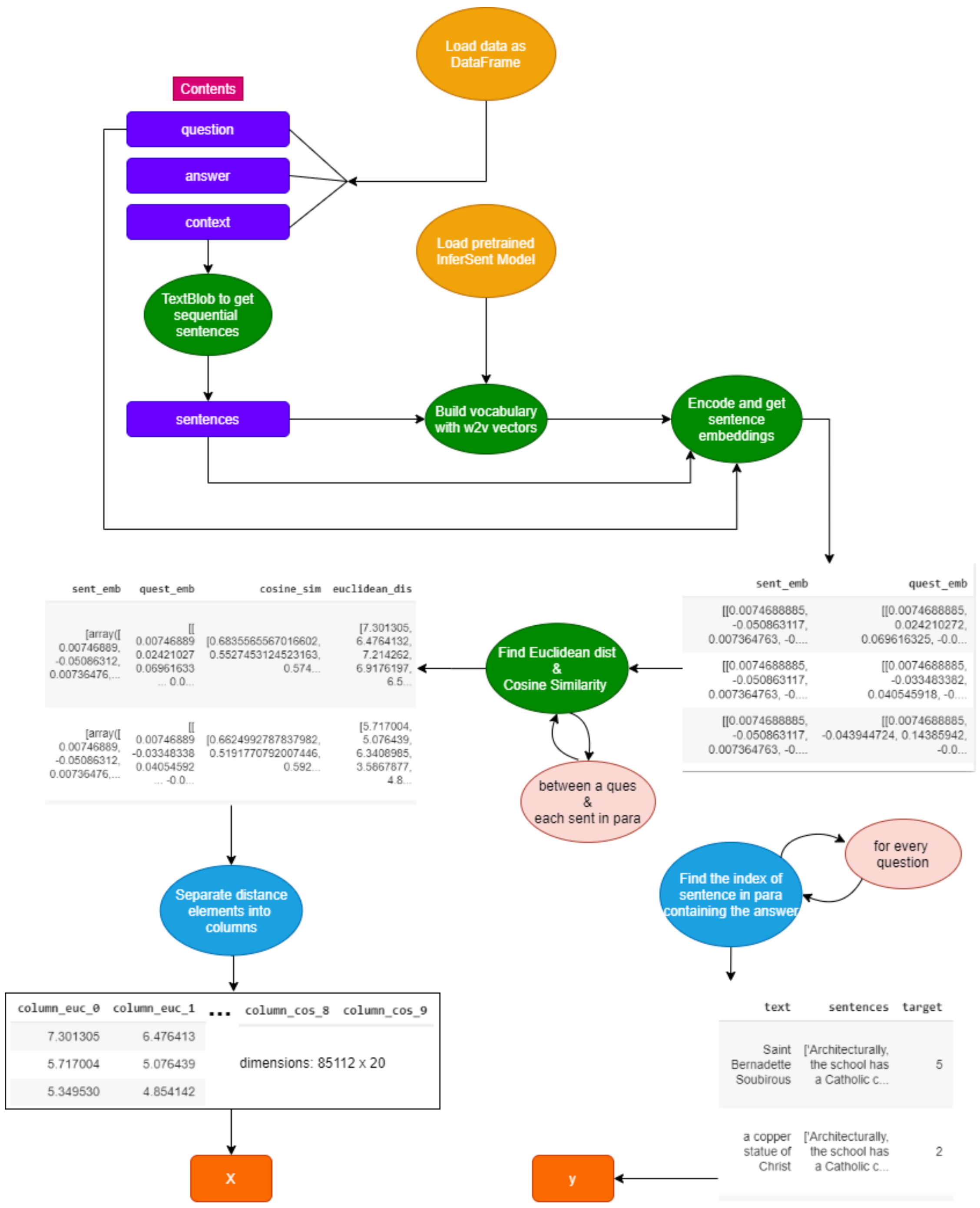
Where do water droplets collide with ice crystals to form precipitation?
within a cloud

  For each observation in the training set, we have a context,question, and a text.The goal is to find the text for any new question and context provided.This is a closed form dataset meaning that the answer to a question is always a part of the context and also a continuous span of context.The above mentioned goal can be achieved by locating the correct sentence in the passage and then further finding the correct text in that located sentence.

- **Data Pre-processing**

  The SQuAD v1.1 dataset is available in Json format and had to be converted to a labelled DataFrame format to use it for processing. After unpacking it, there were 4 columns (namely: context (the paragraph), question, text (the desired answer), and answerstart (character index in the para where the answer begins)) and such 85,112 samples.We use TextBlob to separate the sentences; TextBlob does this while maintaining the sequence of sentences and preserving the content as it is.

  To generate sentence embeddings, we use Facebook research's InferSent which is a method that provides semantic representations (sentence embeddings) for English sentences. Being pretrained on a larger corpus and after building a preliminary vocabulary on SQuAD, we can get contextual embeddings with stronger weights to the more important/significant words of the sentence. From the distance and cosine similarity between sentences and questions, visualised in the multidimensional vector space, we can get an idea about the sentence that most closely gives the answer. Further, we attempted to use ML algorithms to determine the best fitting sentence embedding. The flowchart below shows the data preprocessing process.



Load data as DataFrame
Contents
question
answer
context
Load pretrained InferSent Model
TextBlob to get sequential sentences
sentences
Build vocabulary with w2v vectors
Encode and get sentence embeddings
Find Euclidean dist & Cosine Similarity
between a ques & each sent in para
Separate distance elements into columns
Find the index of sentence in para containing the answer
for every question
x
y

sent_emb	quest_emb
[[0.0074688885, -0.050863117, 0.007364763, -0....	[[0.0074688885, 0.024210272, 0.069616325, -0.0...
[[0.0074688885, -0.050863117, 0.007364763, -0....	[[0.0074688885, -0.033483382, 0.040545918, -0....
[[0.0074688885, -0.050863117, 0.007364763, -0....	[[0.0074688885, -0.043944724, 0.14385942, -0.0...

sent_emb	quest_emb	cosine_sim	euclidean_dis
[array([0.00746889, -0.05086312, 0.00736476,...	[[0.00746889 0.02421027 0.06961633 ... 0.0...	[0.6835565567016602, 0.5527453124523163, 0.574...	[7.301305, 6.4764132, 7.214262, 6.9176197, 6.5...
[array([0.00746889, -0.05086312, 0.00736476,...	[[0.00746889 -0.03348338 0.04054592 ... -0.0...	[0.6624992787837982, 0.5191770792007446, 0.592...	[5.717004, 5.076439, 6.3408985, 3.5867877, 4.8...

column_euc_0	column_euc_1	...	column_cos_8	column_cos_9
7.301305	6.476413			
5.717004	5.076439			
5.349530	4.854142			

dimensions: 85112 x 20

text	sentences	target
Saint Bernadette Soubirous	['Architecturally, the school has a Catholic c...	5
a copper statue of Christ	['Architecturally, the school has a Catholic c...	2

Fig 1. Data Preprocessing

3 Machine Learning Concept Used

- **Logistic Regression**

  Logistic Regression is a supervised classification algorithm which is used to classify the predictions on the data into classes. In our project, we have implemented logistic regression in place of linear regression since the target variable is text and also the algorithm uses MLE estimation which can work well with categorical data and help in better classification.

  Mathematics for logistic regression:

  Let the data be in the form (X,Y) where X is a matrix of m values and n features and Y is a vector of m examples. In our case, X is (85112 x 20) and Y i (85112 x 1). There will be n+1 weights (including the dummy).

  Now, a weight matrix is created randomly and then multiplied with the features.

  $$z = w_0 + w_1 * x_1 + w_2 * x_2 + ...w_n * x_n$$

  Next, a sigmoid function also known as link function will take the input z and generate output $\hat{y_i}$.

  $$\hat{y_i} = \frac{1}{1+e^{-z}}$$

  Cost for an iteration is calculated using log-likelihood function as shown:

  $$cost(w) = (\frac{-1}{m})\Sigma_{i=1}^{m}(y_i log(\hat{y_i}) + (1 - y_i)log(1 - \hat{y_i}))$$

  The equation for the derivative of the cost will be:

  $$dw_j = \Sigma_{i=1}^{n}(\hat{y} - y){x_j}^i$$

  Gradient descent algorithm is used to update the weight parameters by minimising the cost function:

  $$w_i = w_j - (\alpha * dw_j)$$

- **Principal Component Analysis (PCA):**

  Principal component analysis is a technique for feature extraction — so it combines our input variables in a specific way, at which point we can drop the least important variables while still retaining the most valuable parts of all of the variables. PCA results in developing new features that are independent of one another.

  In our project the dataset contains 20 features. The matrix X is multiplied with its transpose in order to find the covariance matrix S of dimensions 20x20. After that, eigenvalues and eigenvectors are calculated for the covariance matrix. We map the 20-dimensional space to a reduced space of 10 dimensions; meaning 10 principal components. From the covariance matrix, projection matrix is formed of shape 10x10 using 10 eigenvectors corresponding to the 10 largest eigenvalues.

Feature Normalization:-

The data normalization is important part for implementation of PCA. The reason for this is that if the data is not normalized then the components we receive after applying PCA might be misleading. We can also simply use a correlation matrix instead of using a covariance matrix if features are of different scales. Here we perform z-normalization.

$$X_{norm} = \frac{X - \mu}{\sigma} \tag{1}$$

The covariance matrix, S, is given by:

$$S = \frac{1}{N}\Sigma_{n=1}^{N} X_n X_n^T \tag{2}$$

where $X_n \epsilon R^D$ and $S \epsilon R^{DXD}$

Here, D = 20

Furthermore, we assume there exists a low-dimensional compressed representation (code):

$$Z_n = B^T X_n \epsilon R^M \tag{3}$$

where B is the projection matrix

$$B := [b_1, ..., b_M] \epsilon R^{DXM} \tag{4}$$

Here, we assume that B is an orthonormal matrix.

The projection matrix B is formed by choosing M eigenvectors (principal components) corresponding to M largest eigenvalues.

The reconstructed matrix will be formed from the multiplication of Projection matrix and Transpose of X.

The reconstructed matrix is given by:

$$\tilde{X} = BB^T X \tag{5}$$

The reconstruction error is given by:

$$J = \frac{1}{N}\Sigma_{n=1}^{N} |X_n - \tilde{X_n}|^2 \tag{6}$$

On increasing the number of components, the reconstruction error reduces.

Advantages of PCA:

1. Eradication of correlated features
2. Improves algorithm performance
3. Reduces overfitting
4. Improves visualization

Disadvantages of PCA:

1. Less interpretable
2. Data standardization is necessary
3. Loss of Information

- **Gaussian Mixture Models**

  Gaussian Mixture Models are probabilistic models which can cluster the data points according to their probability distributions. Data points having a single distribution are grouped together. The clustering approach utilises both mean and variance and hence the results are more accurate.

To begin with, a Gaussian mixture is a function which comprises of several gaussians identified by k ∈ 1,...,K where K is the number of clusters for the dataset. Each gaussian k comprises of - mean (μ), variance (Σ), mixing probability (π).
The mixing coefficients are probabilities and therefore the following condition should be met.

$$\Sigma_{k=1}^{K}\pi_k = 1$$

Gaussian density functions is given as:

$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} exp(-1/2(x-\mu)^T\Sigma^{-1}(x-\mu))$$

x = data points
D = dimension of data points
By combining multiple simpler models into a complex model,

$$p(x) = \Sigma_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)$$

The likelihood of GMM for N IID points of dataset is:

$$\Pi_{i=1}^{N}p(x_i) = \Pi_{i=1}^{N}\Sigma_{k=1}^{K}\pi_k\mathcal{N}(x_i|\mu_k,\Sigma_k)$$

The cost function is defined as negative of log-likelihood:

$$L(\theta) = -\Sigma_{i=1}^{N} ln\Sigma_{k=1}^{K}\pi_k\mathcal{N}(x_i|\mu_k,\Sigma_k)$$

θ refers to the parameters - μ_k and Σ_k.
To minimise $L(\theta)$ we first differentiate with respect to μ_k and then with Σ_k.

$$\mu_k = \frac{1}{N_k}\Sigma_{i=1}^{N}\gamma_{ik}x_i \text{ (weighted mean)}$$
$$\Sigma_k = \frac{1}{N_k}\Sigma_{i=1}^{N}\gamma_{ik}(x_i-\mu_k)(x_i-\mu_k)^T \text{ (weighted variance)}$$

Now, in order to minimise the cost function - Expectation Maximization (EM) algorithm is used.

1. Expectation : Responsibilities are calculated using current parameters.

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k,\Sigma_k)}{\Sigma_{j=1}^{K}\pi_j \mathcal{N}(x_i|\mu_j,\Sigma_j)}$$

2. Maximization : Update the parameters by using responsibilities.

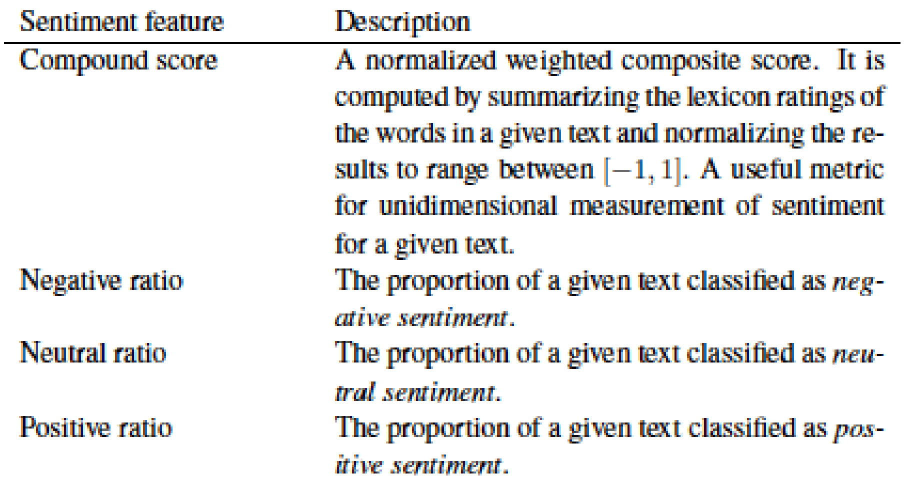
$$\mu_k = \frac{1}{N_k}\Sigma_{i=1}^{N}\gamma_{ik}x_i$$
$$\Sigma_k = \frac{1}{N_k}\Sigma_{i=1}^{N}\gamma_{ik}(x_i - \mu_k)(x_i - \mu_k)^T$$
$$\pi_k = \frac{N_k}{N} \text{ where } N_k = \Sigma_{i=1}^{N}\gamma_{ik}$$

Above steps are performed till the optimal values of parameters are obtained, i.e. until convergence.

In our project, we have a data matrix of 85119*20 dimension.The 20 features include 10 columns of euclidean distance and 10 columns of cosine similarity which are normalised using minmax scalar. Since, the gaussian distribution is plotted against 2 features, we have selected column 1 from the euclidean distance and corresponding column 1 from the cosine similarity. Our dataset is classified into 10 classes, ranging from 0-9, thus the number of components/clusters are chosen to be 10 in number.

- **Sentiment Analysis**
  Sentiment Analysis is a natural language processing technique used to identify and study the affective state or subjective information in texts.This technique is widely used for interpreting user reviews, analysing social media,etc. Sentiment polarity classification is a binary classification task for labeling a text document as positive or negative.In our project,we have classified our paragraphs into either positive,negative or neutral.

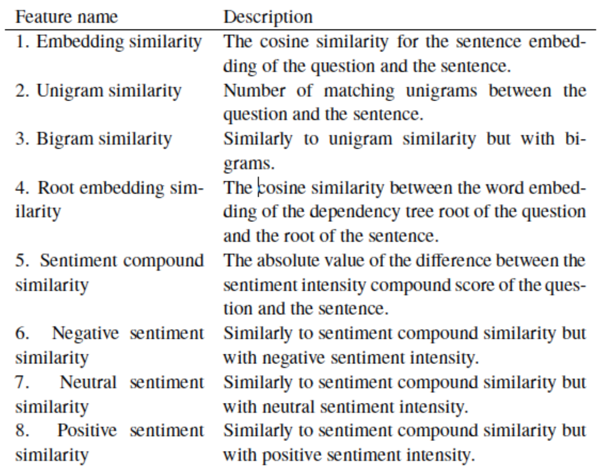
  Sentiment analysis relies on sentiment lexicons. A sentiment lexicon is a list of words and lexical features which are generally labelled according to their semantic values. We have used VADER rule based model which is based on social media texts. The VADER algorithm provides 4 different measurements of sentiment intensity as shown below.



Sentiment feature	Description
Compound score	A normalized weighted composite score. It is computed by summarizing the lexicon ratings of the words in a given text and normalizing the results to range between $[-1, 1]$. A useful metric for unidimensional measurement of sentiment for a given text.
Negative ratio	The proportion of a given text classified as *negative sentiment*.
Neutral ratio	The proportion of a given text classified as *neutral sentiment*.
Positive ratio	The proportion of a given text classified as *positive sentiment*.

Table-1

In order to perform sentiment analysis, there were several steps of pre-processing to build vectors. The articles and questions of the dataset were tokenised and the sentences were split into candidate answers. We used Word2vec group of models for converting word into numerical vectors known as embeddings.This type of representation is robust and straightforward, but if the words are large enough, the vectors quickly become very sparse, which makes it inefficient.An answer vector was represented using the features[1-4] which were used to measure the semantic similarity between a question and a sentence,while features[5-8] were used to measure sentiment similarity.The sentiment intensities were extracted using VADER.



Feature name	Description
1. Embedding similarity	The cosine similarity for the sentence embedding of the question and the sentence.
2. Unigram similarity	Number of matching unigrams between the question and the sentence.
3. Bigram similarity	Similarly to unigram similarity but with bigrams.
4. Root embedding similarity	The cosine similarity between the word embedding of the dependency tree root of the question and the root of the sentence.
5. Sentiment compound similarity	The absolute value of the difference between the sentiment intensity compound score of the question and the sentence.
6. Negative sentiment similarity	Similarly to sentiment compound similarity but with negative sentiment intensity.
7. Neutral sentiment similarity	Similarly to sentiment compound similarity but with neutral sentiment intensity.
8. Positive sentiment similarity	Similarly to sentiment compound similarity but with positive sentiment intensity.

Table-2

When we applied sentiment analysis on the paragraphs we found that a particular sentence in a paragraph either belongs to positive, negative or neutral sentiment. In case, when no particular sentiment was present , percentage of confidence for every sentiment was obtained.

- **XGBoost Algorithm**

  XGBoost is a decision tree algorithm based on ensemble learning method. Ensemble learning will enable to produce combined output of multiple models by using predictive power of all. Bagging and boosting are used as ensemble learners. While bagging involves presence of several trees in parallel and then generating the output by averaging over all the learners. On the other hand, boosting generates trees sequentially so that each subsequent tree will diminish the errors of the previous trees.

  XGBoost gives better performance than random forest because random forest utilises fully grown decision trees whereas the former works on trees with fewer splits which reduces error.

  Mathematics for Boosting ensemble:

  - A model is intialized with a function F0, whose predicted target value is y. The residual value is calculated as (y-F0).

    $$F_o(x) = argmin_y \Sigma_{i=1}^{n} L(y_i, \gamma)$$
    $$\text{where, } argmin_y \Sigma_{i=1}^{n} L(y_i, \gamma) = argmin_y \Sigma_{i=1}^{n} (y_i - \gamma)^2$$

    Here,$L(y_i, \gamma)$ is the mean-squared error loss function.

  - The loss function will be minimised and the value of multiplicative factor α is determined using gradient descent algorithm. The pseudo residual is calculated as follows:

    $$r_{im} = -\alpha \left[\frac{\partial L(y_i, F(x_i))}{\partial (F(x_i))} \right]_{F(x)=F_{m-1}(x)}$$

  - Now, a new model is defined as hn which will fit residuals from the previous trees. For the gradient acquired at each step, hn(x) is fit to it.

  - α_n is the multiplicative factor derived for each terminal node and the boosted model Fn(x) is defined:

    $$F_n(x) = F_{n-1}(x) + \alpha_n h_n(x)$$

    $F_n(x)$ is the minimisation of residuals after n iterations.

- **Random Forest:**

  Random forests (RF) build different decision trees. Predictions from all trees are brought together to make the final prediction; the mode of the classes for classification or the mean prediction for regression. The decisions trees are then trained individually, and their predictions are used to output final prediction. The predictions of all decision trees are brought together and the method to do this is called as Ensemble Methods.

  Implementation:

  For each decision tree, calculating node importance using Gini Importance, assuming only two child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \tag{7}$$

ni(j)= the importance of node j

w(j) = weighted number of samples reaching node j

C(j)= the impurity value of node j

left(j) = child node from left split on node j

right(j) = child node from right split on node j

The importance for each feature on a decision tree is then calculated as:

$$fi_i = \frac{\Sigma_{j:nodejsplitsonfeaturei} ni_j}{\Sigma_{k\epsilon allnodes} ni_k} \tag{8}$$

fi(i)= the importance of feature i

ni(j)= the importance of node j

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$RFfi_i = \frac{\Sigma_j normfi_{ij}}{\Sigma_{i\epsilon allfeatures, k\epsilon alltrees} normfi_{jk}} \tag{9}$$

RFfi(i)= the importance of feature i calculated from all trees in the Random Forest model

normfi(ij)= the normalized feature importance for i in tree j

Support Vector Machines:-

SVM a supervised learning algorithm. It is used for both classification and regression task.

Hyperplane:- It is a plane that straightly partitions the n-dimensional information focuses in two-segment. On account of 2D, a hyperplane is a line, in 3D it is plane.It is additionally called a n-dimensional line.

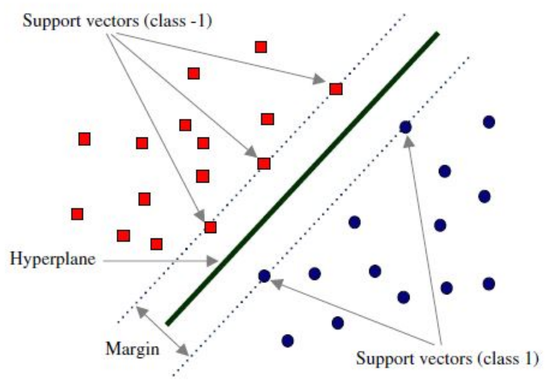
$$y = ax + b \tag{10}$$

$$ax + b - y = 0 \tag{11}$$

X=(x,y) and W=(a,-1)

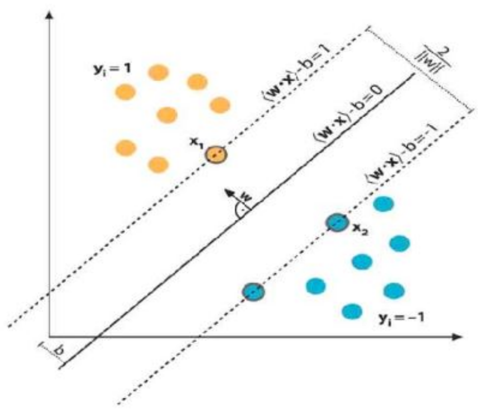
$$W.X + b = 0 \tag{12}$$

Optimal Hyperplane:-



So, In SVM our goal is to choose an optimal hyperplane which maximizes the margin.

Mathematical Analysis of SVM:-


$y_i = 1$
$\langle w \cdot x \rangle - b = 1$
$\langle w \cdot x \rangle - b = 0$
$\langle w \cdot x \rangle - b = -1$
$\frac{2}{\|w\|}$
x_1
x_2
w
b
$y_i = -1$

Support vectors models are nearest to ideal hyperplane and the point of the SVM is to orientate this hyperplane quite far from the nearest individual from the two classes.

SVM Formulation:-

w.x+b≥1 for y=+1 class

w.x+b≤-1 for y=-1 class

Above two equation can be written as:

y(w.x+b)+1≥0, for y=+1 and -1 class.

We have two hyperplane H1 and H2 going through the support vectors of +1 and - 1 class individually.

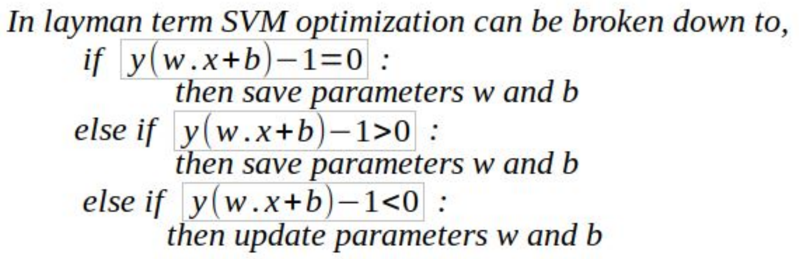
w.x+b=-1 :H1

w.x+b=1 :H2

What's more, separation between H1 hyperplane and starting point is $(-1 - b)/|w|$ also, separation between H2 hyperplane and source is $(1–b)/|w|$. Margin is as follows:

$M = 2/|w|$

Optimization:



```
In layman term SVM optimization can be broken down to,
    if $\boxed{y(w.x+b)-1=0}$ :
        then save parameters w and b
    else if $\boxed{y(w.x+b)-1>0}$ :
        then save parameters w and b
    else if $\boxed{y(w.x+b)-1<0}$ :
        then update parameters w and b
```

In our project, we have implemented support vector machine using different kernels.Using default SVC function, we got a matrix of support vectors with dimension (52295x20). Upon implementing linear kernel, it performs closer to the logistic regression approach whereas polynomial and gaussian kernel perform better upon tuning the values of C and gamma as hyperparameters. On using GridSearchCV and cross-validation methods, best hyperparameters for implementing SVM can be obtained.

4 Pseudo Code/ Algorithm

Algorithm 1 Gaussian Mixture Model

```
    Initialize the mean $\mu_k$, the covariance matrix $\Sigma_k$ and the mixing coefficients $\pi_k$ by some random values.
    Compute the $\gamma_k$ values for all k.
    Again Estimate all the parameters using the current $\gamma_k$ values.
    Compute log-likelihood function. Put some convergence criterion if the log-likelihood value converges to some value
    then stop else GO TO STEP 2
```

Algorithm 2 Principal Component Analysis

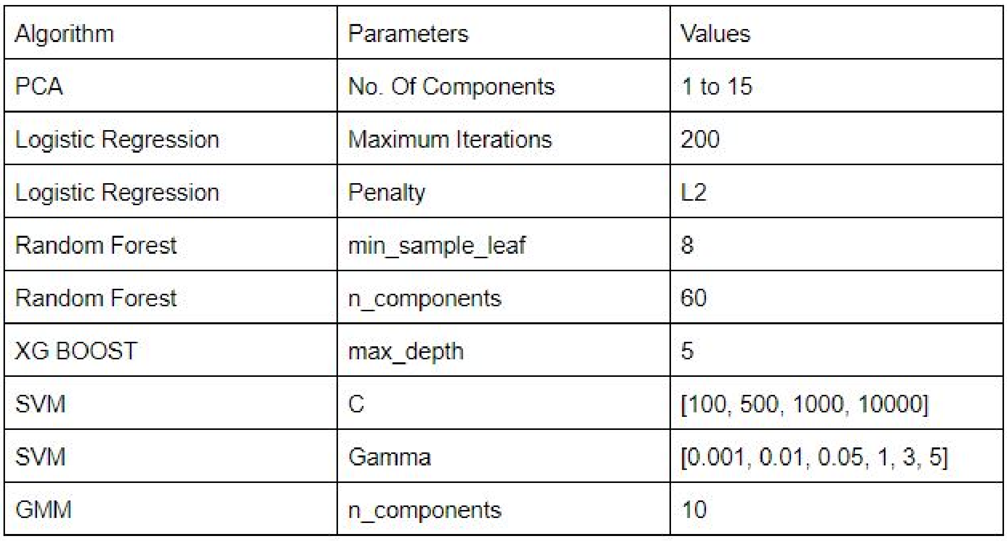
```
    Input: a D-dimensional training set X from $x_1$ to $x_n$ and the new lower dimensionality d less than equal D
    Compute the mean $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
    Compute the covariance matrix Cov(x) $= \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x_i})(x_i - \bar{x_i})^T$
    Find the spectral decomposition of Cov(x),obtaining the eigen vectors $\zeta_1, \zeta_2, .....\zeta_n$ and their corresponding eigen values $\lambda_1, \lambda_2, ....\lambda_n$.Note that the eigen values are sorted and arranged in a descending order
    For any $x \in R^D$ ,its new lower dimensional representation is: $y = (\zeta_1^T(x-\bar{x}), \zeta_2^T(x-\bar{x}), ...., \zeta_d^T(x-\bar{x}))^T \in R^D$
    and the original x can be approximated as $x = \bar{x} + (\zeta_1^T(x-\bar{x})\zeta_1 + \zeta_2^T(x-\bar{x})\zeta_2 + .... + \zeta_d^T(x-\bar{x})\zeta_d)$
```

Algorithm 3 Logistic Regression

```
    Initialize: $\theta_j = 0$ for all $0 \le j \le m$
    Repeat Many Times:
    gradient[j]=0 for all $0 \le j \le m$
    For each training example (x,y):
    For each parameter j:
    gradient[j] += $x_j(y - \frac{1}{1+e^{-\theta^T x}})\theta_j +\ = \eta * gradient[j]\ for\ all\ 0 \le j \le m$
```

5 Coding and Simulation

5.1 Simulation Framework

1: 

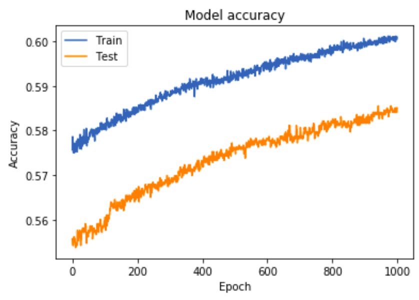
Algorithm	Parameters	Values
PCA	No. Of Components	1 to 15
Logistic Regression	Maximum Iterations	200
Logistic Regression	Penalty	L2
Random Forest	min_sample_leaf	8
Random Forest	n_components	60
XG BOOST	max_depth	5
SVM	C	[100, 500, 1000, 10000]
SVM	Gamma	[0.001, 0.01, 0.05, 1, 3, 5]
GMM	n_components	10

Simulation Parameters used

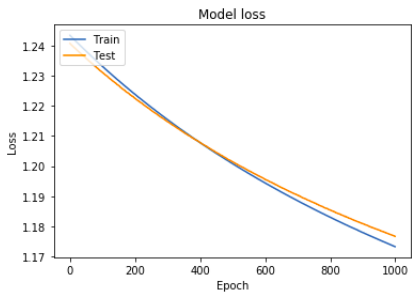
5.2 Results

- **Logistic Regression**

  Plots 1 and 2 shows the result obtained for logistic regression without regularisation. It can be observed that testing data accuracy starts little bit below than the training data whereas the loss for training and data curves are almost similar.

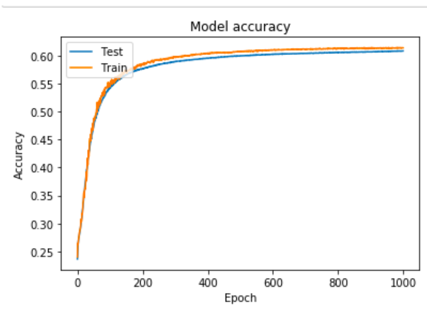


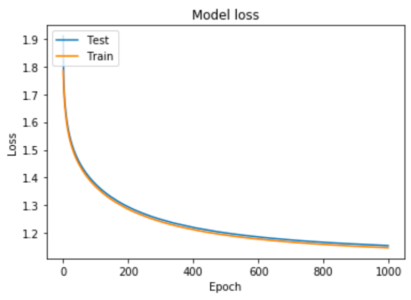
Plot:1 Accuracy



Plot:2 Loss

  Plots 3 and 4 shows the result obtained for logistic regression with regularisation.It can be observed that after 200 epochs no major change can be observed in the accuracy plot for training and testing data.Whereas the training loss decreases with increasing number of epochs.



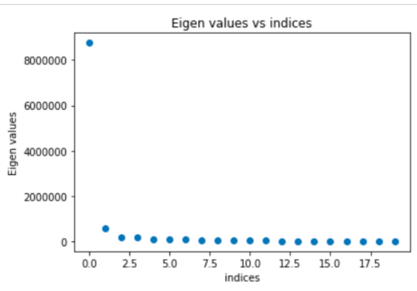


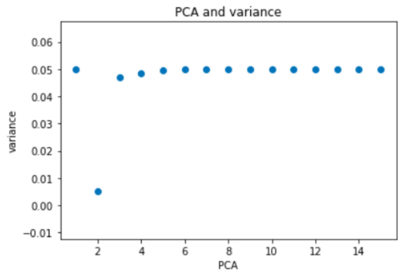
Plot:1 Accuracy Plot:2 Loss

- **Principal Component Analysis**

  Plot 5 represents trend between eigen values and its indices.The eigen values are sorted in descending order. In our case, the eigen values are nearly of similar values except the first eigen value.On arranging the eigen values in an unsorted manner, mean square error changes slightly.

  Plot 6 shows the trend between the principal components and the variance.The components represent the reduced version of euclidean distance and the cosine similarity between questions and the sentences of the passage which are similar in some cases.Thus,on increasing the number of components,our variance didn't change significantly because of similar components.

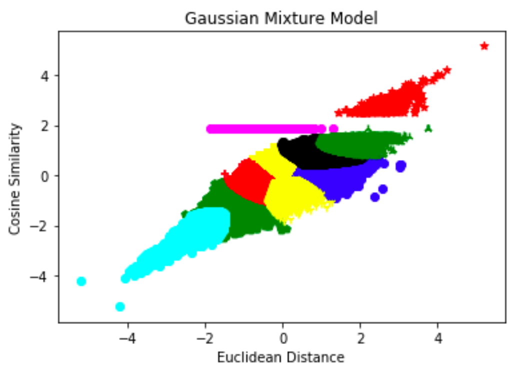




Plot:5 Eigen Values VS Indices Plot:6 Principal Components VS Variance

- **Gaussian Mixture Models**

  Plot 6 represents several clusters for different classified classes(10).The x-axis represents the euclidean distance and the y-axis represents the cosine similarity. Since, our total features were 20, we chose 2 from them- one column from euclidean and another from cosine similarity.The clusters obtained are closer too each other due to similarity between euclidean distances and cosine similarities between sentences of the passages.



Plot:6 Gaussian Mixture Model Representation

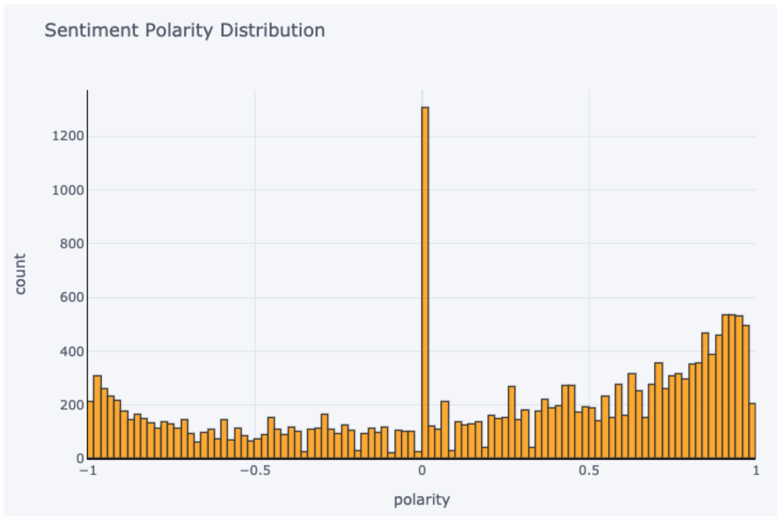
10 clusters for different classes (Note:Red,green and yellow are used repeatedly because RGB combination only yielded 8 different colors excluding white)

5.3 New Models and Approaches used

5.3.1 Results obtained from new models

- **Sentiment Analysis of SQuAD 1.1**

  Plot: 7 shows a sentiment polarity distribution with respect to the frequency of questions having different polarity.We implemented the VADER Sentiment Analyser from the NLTK library on our paragraphs.Around 1300 paragraphs among 19000 are completely neutral.Whereas around 250-300 paras are completely positive and completely negative.

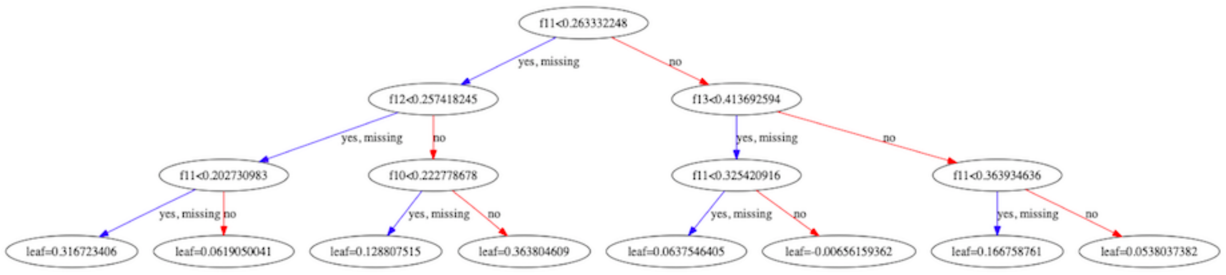


Sentiment Polarity Distribution

- **XGBoost**

  The extreme gradient boosting mechanism helps improve performance as opposed to random forest because it works on trees with fewer splits. The training accuracy is 67% and testing accuracy is 65%

which is lower than random forest. The testing accuracy will be more as we increase the value of maxdepth parameter since it represents the size of decision trees. The tree plot represents the feature indices and their values as per the data given. The nodes are split according to the answers which then become output leaf nodes.



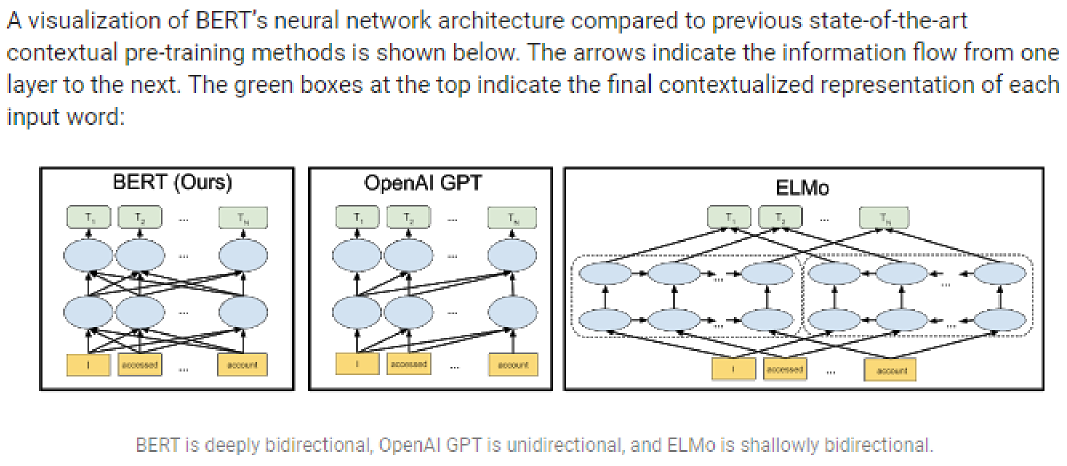
Tree generated using XGBoost

- **Random Forest**

  The decision tree in random forest forms nodes to obtain a large number of data points from a class by finding values in the features which will divide the data samples in classes. The training and testing accuracy depends on the number of samples at leaf nodes and the number of trees. When the values of parameters - minsamplesleaf (samples at leaf node) = 8 and nestimators (number of trees) = 60, training accuracy is 77.8 percent and testing accuracy is 63.7 percent. However, when minsamplesleaf = 3 and nestimators = 5, training accuracy is 85.5 percent and testing accuracy is 58.7 percent. In general, as the value of both the parameters is decreased, training accuracy increases while testing accuracy decreases which denotes that model is overfitting the data.

- **Bidirectional Encoder Representations and Transformer (BERT)**

  Previous language models, especially for generating embedding vectors, used unidirectional training: a context window moved along from either L to R or R to L around the target word which had the potential to miss important context later in the sentence. BERT looks at it bidirectionally and the whole sentence simultaneously in the way that humans look at the whole context of a sentence. Additionally, BERT is pretrained using Masked Language Model (MLM) method which randomly masks tokens and trains on predicting the masked token. This prevents target words in the process from seeing themselves during bidirectional training.



A visualization of BERT's neural network architecture compared to previous state-of-the-art contextual pre-training methods is shown below. The arrows indicate the information flow from one layer to the next. The green boxes at the top indicate the final contextualized representation of each input word:

BERT is deeply bidirectional, OpenAI GPT is unidirectional, and ELMo is shallowly bidirectional.

Fig 2. Bidirectional nature of BERT

BERT utilizes a sum of three types of embedding vectors (tensors) to best represent the sentence along with contextual significance. For example in the following two sentences:

“The man was accused of robbing a bank. The man went fishing by the bank of the river.”

Previous state-of-the-art sentence representations would represent both instances of ”bank” with the same or a nearby vector. Meanwhile, due to its bidirectional nature, BERT would recognize the context of ”bank” in each of the sentences using words like ”river”, ”fishing”, and ”robbing” and similar contexts.

Using three embedding layers namely the Token Embeddings, Segment Embeddings, and the Position Embeddings. Each of these layers return a tensor of dimensions *(batchsize, #tokens, hidden units)* as shown in the flowchart below.

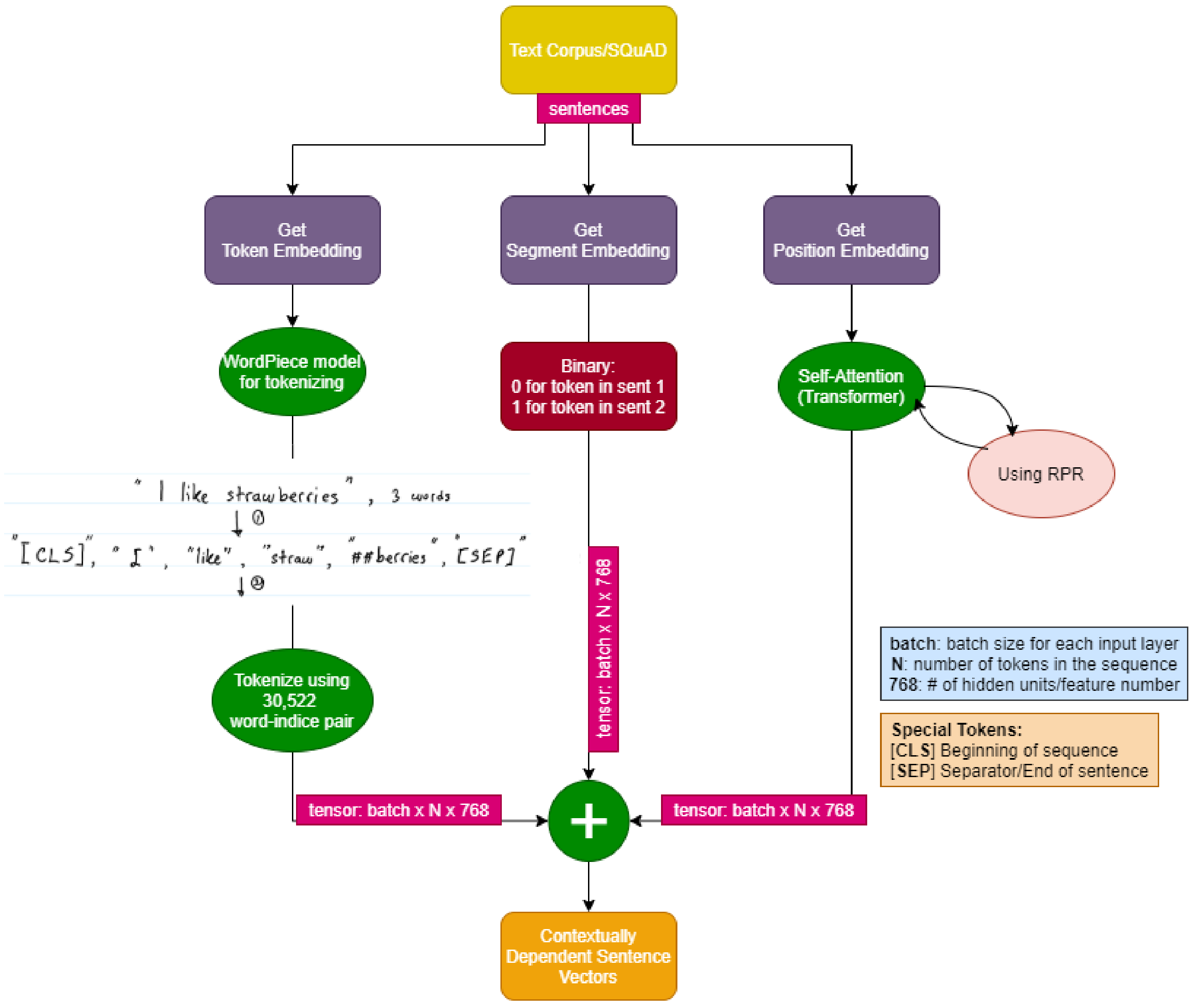

Text Corpus/SQuAD
sentences
Get Token Embedding
Get Segment Embedding
Get Position Embedding
WordPiece model for tokenizing
Binary: 0 for token in sent 1 1 for token in sent 2
Self-Attention (Transformer)
Using RPR
" I like strawberries " , 3 words
①
"[CLS]", " I ", "like", "straw", "##berries", "[SEP]"
②
tensor: batch x N x 768
batch: batch size for each input layer
N: number of tokens in the sequence
768: # of hidden units/feature number
Special Tokens:
[CLS] Beginning of sequence
[SEP] Separator/End of sentence
Tokenize using 30,522 word-indice pair
tensor: batch x N x 768
tensor: batch x N x 768
Contextually Dependent Sentence Vectors

Fig 3. Creating BERT embeddings

The token embeddings (tokenizing) are generated using the encoding block of the Transformer proposed in [3]. As opposed to traditional machine translation (encoding/tokenizing) methods which used RNN/LSTMs, this Transformer employs multi-head self-attention technique. During training, there is an additional set of trainable embeddings (vectors) that are used when computing the attention weight and value (relative distance/ of words) between word i and j in the input sequence. This is called Relative Position Representation (RPR). It helps in representing the sequential information of its input in the output representation.
Thus, a sequence of k words will have a total of 2k+1 embeddings to learn (1 embedding for the current word, k embeddings for the k words to the left and k embeddings for the k words to the right of the target word).

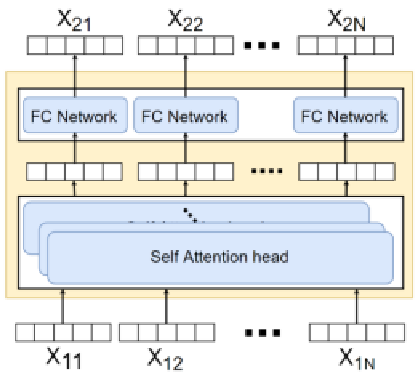


Fig 2: Transformer encoder which consists of self-attention heads and fully connected neural networks. This encoder modifies the representation of each token to suit the contents of the other tokens and presents a new representation. Each self-attention head discovers a new semantic relation between various tokens and converts it into a new vector similar to input vectors using a fully connected neural network.

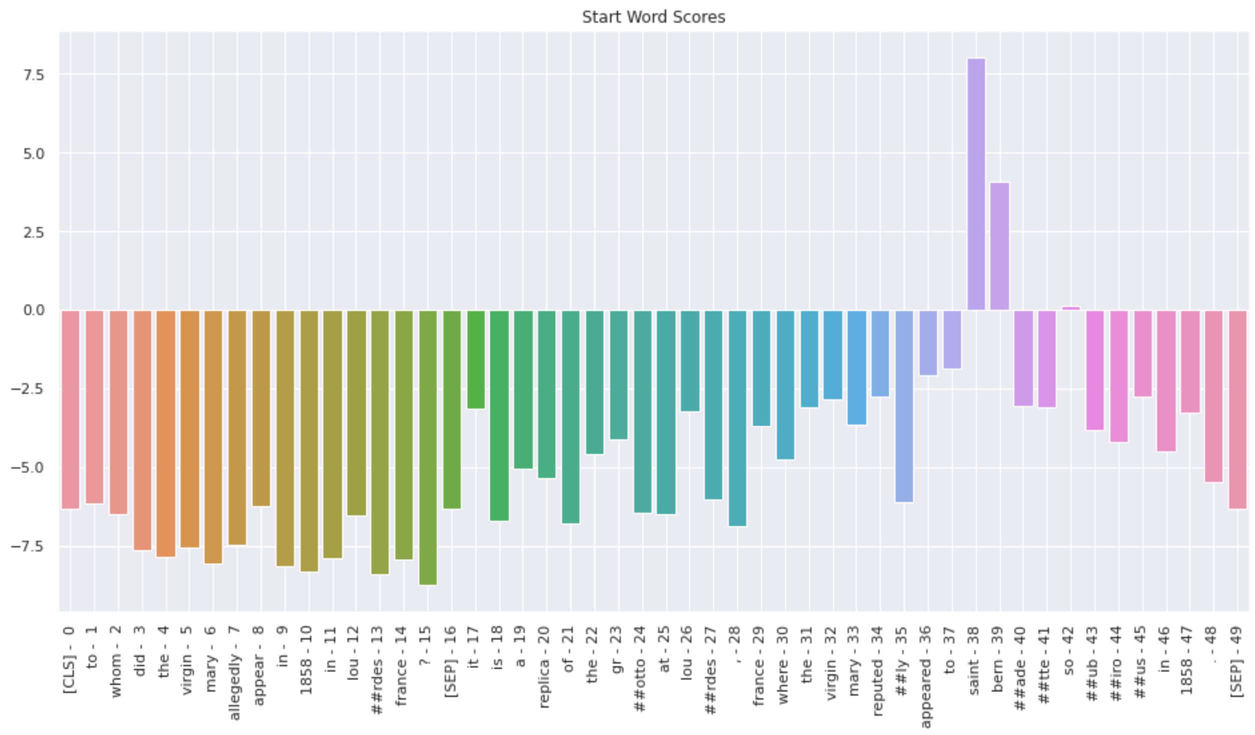
Transformer with FC-NN (Source: Mozafari et. al. 2019 [10])

The model learns new embeddings, given a paragraph and a question, to generate scores for each token in the sequence. The best scores show the tokens which contain the nearest answer to the question. For example in the below paragraph and question taken from SQuAD 1.1:

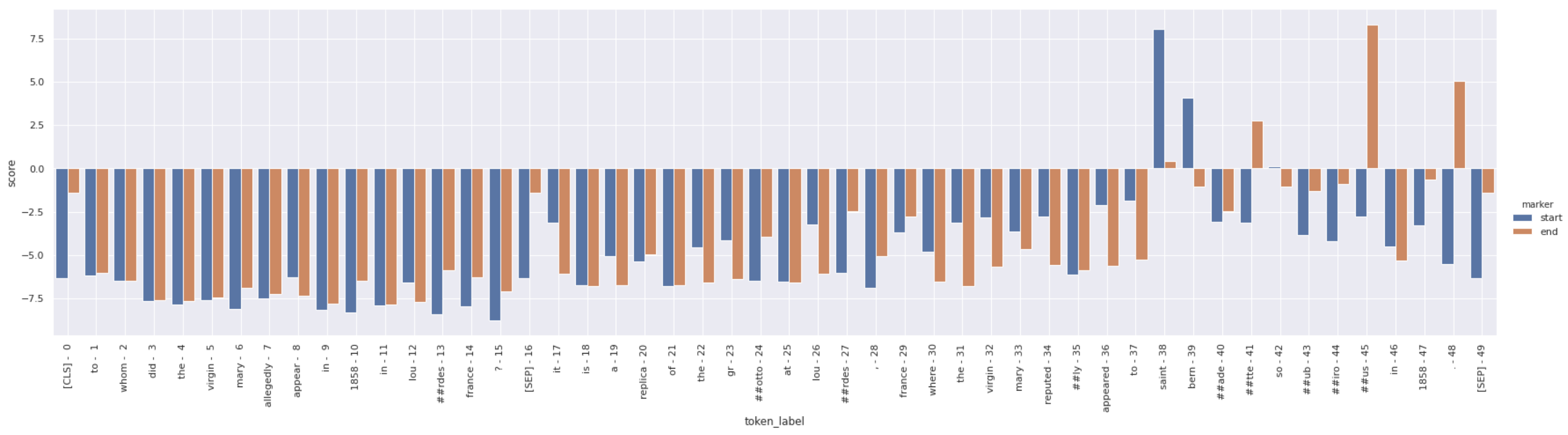
"Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary. Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858. At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary."

Question: *"To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?"*

After putting them through the model, we get scores for "answer start" and "answer end" and the tokens between the best of each of these scores contains the answer. As shown below in the graph, we see that the highest scores form the sequence "Saint Bernadette Soubirous" which happens to be the correct answer.



Scores for ”answer start”



Plot for both scores (answer start and answer end)

5.3.2 Algorithms for new models

Algorithm 4 XGBoost

```
Initialize: $\hat{f(x)} = 0$ and $r_i = y_i$ for all i in training set.    For b = 1,2,...,B, repeat:
Fit a tree $f^b$ with d splits (d + 1 terminal nodes) to the training data (X,r).
Update $\hat{f}$ by adding in a shrunken version of the new tree:    $\hat{f(x)} \leftarrow \hat{f(x)} + \lambda f^b(x)$
Update the residuals,    $r_i \leftarrow r_i - \lambda f^b(x_i)$ Output the boosted model,    $\hat{f(x)} = \Sigma_{b=1}^{B} \lambda f^b(x)$
```

Algorithm 5 Random Forest

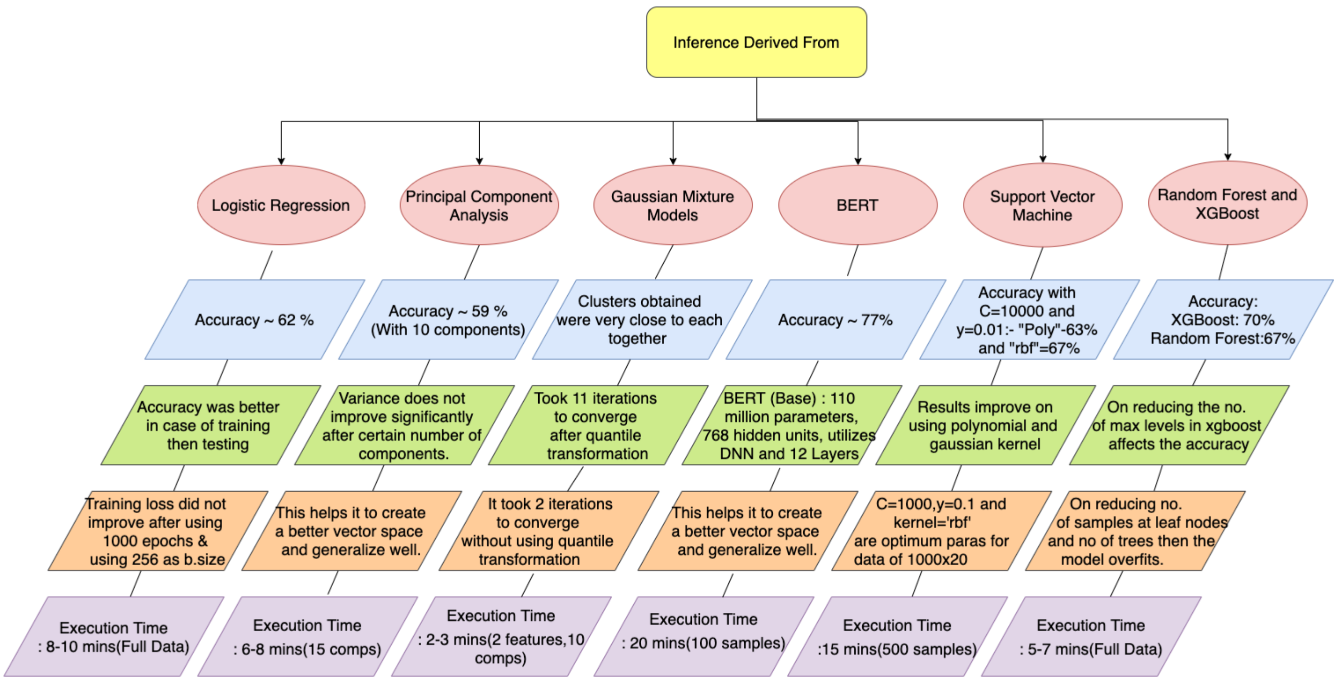
```
for 1 to T  Draw n points $D_l$ with replacement from D   Build full decision/regression tree on $D_l$
BUT: each split only consider k features, picked uniformly at random new features for every split
Prune tree to minimise out-of-bag error Average all T trees
```

Algorithm 6 Support Vector Machine

```
Initialize a two dimensional plane: y = ax + b    Let X = (x,y) and W = (a,-1)
Vector form hyperplane → W.X + b = 0    Hyperplane 1 → w.x + b = -1
Hyperplane 2 → w.x + b = 1IF y(w.x + b) -1 = 0 then save parameters w and b ELSE IF
y(w.x + b) -1 > 0 then save parameters w and b ELSE update parameters w and b
```

6 Inferences/Conclusions

After applying various ML models and BERT to our dataset, we were able to derive the following inferences (as shown in the diagram below). Some models being too computational intensive for our PCs, we had to truncate our dataset to get a representative result.



Inference Derived From
Logistic Regression
Principal Component Analysis
Gaussian Mixture Models
BERT
Support Vector Machine
Random Forest and XGBoost
Accuracy ~ 62 %
Accuracy ~ 59 % (With 10 components)
Clusters obtained were very close to each together
Accuracy ~ 77%
Accuracy with C=10000 and y=0.01:- "Poly"-63% and "rbf"=67%
Accuracy: XGBoost: 70% Random Forest:67%
Accuracy was better in case of training then testing
Variance does not improve significantly after certain number of components.
Took 11 iterations to converge after quantile transformation
BERT (Base) : 110 million parameters, 768 hidden units, utilizes DNN and 12 Layers
Results improve on using polynomial and gaussian kernel
On reducing the no. of max levels in xgboost affects the accuracy
Training loss did not improve after using 1000 epochs & using 256 as b.size
This helps it to create a better vector space and generalize well.
It took 2 iterations to converge without using quantile transformation
This helps it to create a better vector space and generalize well.
C=1000,y=0.1 and kernel='rbf' are optimum paras for data of 1000x20
On reducing no. of samples at leaf nodes and no of trees then the model overfits.
Execution Time : 8-10 mins(Full Data)
Execution Time : 6-8 mins(15 comps)
Execution Time : 2-3 mins(2 features,10 comps)
Execution Time : 20 mins(100 samples)
Execution Time :15 mins(500 samples)
Execution Time : 5-7 mins(Full Data)

We can conclude that despite using InferSent for generating contextual sentence representations, simple ML models could not perform well (in most cases we observed overfitting or poor performance despite increasing data samples or epochs). Comparing the results with a model like BERT, we can conclude that for comprehensive sentence representations larger and more complex models that employ DL techniques work better. Different versions of BERT such as BERT-Large, Multilingual-BERT, ALBERT (currently SOTA), RoBERTa, etc. can be studied to derive better inferences and results on SQuAD.

7 Contribution of team members

7.1 Technical contribution of all team members

Tasks	Yesha Shastri	Ratnam Parikh	Devshree Patel	Param Raval
1.	Generation of embeddings	Sentiment Analysis on data	Data Pre-processing	Dataset and Lit-Review
2.	XGBoost, LR	Random Forest, PCA	SVM, GMM	BERT
3.	PseudoCode	Algorithms, Simulation	Plotting Respective graphs	Plotting Respective graphs

7.2 Non-Technical contribution of all team members

Tasks	Yesha Shastri	Ratnam Parikh	Devshree Patel	Param Raval
1.(Report)	Inferences	Background	Inferences and Flowchart(Concl.)	Motivation and Flowcharts(BERT)
2.	Exploring new models	Exploring new models	Debugging	Debugging
3.	Integration and Co-ordination	Responsibility and Communication	Leadership and Integration	Initiative and Co-ordination

References

[1] Rajpurkar, Pranav, et al. ”SQuAD: 100,000+ questions for machine comprehension of text.” arXiv preprint arXiv:1606.05250 (2016)

[2] Vaswani, Ashish, et al. ”Attention is all you need.” Advances in neural information processing systems. 2017.

[3] Devlin, Jacob, et al. ”BERT: Pre-training of deep bidirectional transformers for language understanding.” arXiv preprint arXiv:1810.04805 (2018).

[4] Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani. ”Self-attention with relative position representations.” arXiv preprint arXiv:1803.02155 (2018).

[5] Transformer: A Novel Neural Network Architecture For Language Understanding https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

[6] Open Sourcing BERT: State-of-the-art Pre-training For Natural Language Processing https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

[7] A Deep Dive Into Bert: How Bert Launched a Rocket Into Natural Language Understanding Dawn Anderson-Dawn Anderson-Dawn Anderson-Dawn Anderson- SEO Search Digital Marketing Strategist-SEO Search Digital Marketing Strategist - https://searchengineland.com/a-deep-dive-into-bert-how-bert-launched-a-rocket-into-natural-language-understanding-324522

[8] BERT Word Embeddings Tutorial http://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

[9] Conneau, Alexis, et al. ”Supervised learning of universal sentence representations from natural language inference data.” arXiv preprint arXiv:1705.02364 (2017).

[10] Mozafari, Jamshid, Afsaneh Fatemi, and Mohammad Ali Nematbakhsh. ”BAS: An Answer Selection Method Using BERT Language Model.” arXiv preprint arXiv:1911.01528 (2019).

[11] Gatt, Albert, and Emiel Krahmer. ”Survey of the state of the art in natural language generation: Core tasks, applications and evaluation.” Journal of Artificial Intelligence Research 61 (2018): 65-170.

[12] Tranaeus, David. ”Influence of Sentiment in Question Answering Systems.” (2019).

[13] Dong, Li, et al. ”Unified language model pre-training for natural language understanding and generation.” Advances in Neural Information Processing Systems. 2019.

[14] How To Visualize a Decision Tree from a Random Forest in Python Using Scikit-learn Will Koehrsen - https://towardsdatascience.com/how-to-visualize-a-decision-tree-from-a-random-forest-in-python-using-scikit-learn-38ad2d75f21c

[15] Math Behind Logistic Regression Algorithm Sidharth Sekhar - https://medium.com/analytics-vidhya/logistic-regression-b35d2801a29c

[16] What Are Gaussian Mixture Models? A Powerful Clustering Algorithm Aishwarya Singh - https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/

[17] Understanding the Math Behind the Xgboost Algorithm Guest Blog - https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

[18] Ng, Hwee Tou, Leong Hwee Teo, and Jennifer Lai Pheng Kwan. ”A machine learning approach to answering questions for reading comprehension tests.” Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13. Association for Computational Linguistics, 2000.

[19] Williams, Adina, Nikita Nangia, and Samuel R. Bowman. ”A broad-coverage challenge corpus for sentence understanding through inference.” arXiv preprint arXiv:1704.05426 (2017).

[20] Turian, Joseph, Lev Ratinov, and Yoshua Bengio. ”Word representations: a simple and general method for semi-supervised learning.” Proceedings of the 48th annual meeting of the association for computational linguistics. Association for Computational Linguistics, 2010.

[21] Mnih, Andriy, and Geoffrey E. Hinton. ”A scalable hierarchical distributed language model.” Advances in neural information processing systems. 2009.