Answering Product-Related User Queries using Capsule Network

Capsule networks are used for understanding the spatial information, as they help in making sense of words even if they are not adjacent, which makes the approach not one of the words to word analysis but provides the classifier to understand the context of the entire network.
A capsule consists of a group of neurons along with the activation, which is the vector representation of instantiation parameters which helps in providing the probability for the existence of an entity in a piece of textual information.
Capsule network uses the dynamic routing algorithm which helps in preserving not only one, but all the useful features for a sentence or a part of the text.

MODEL

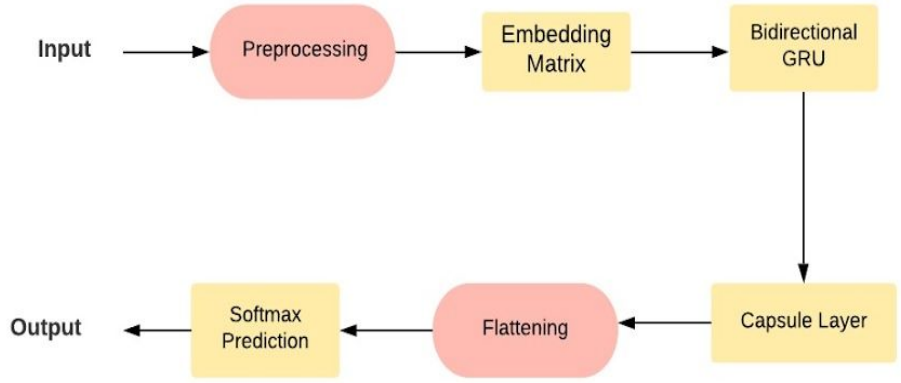
Dataset

The dataset is incorporated from the Flipkart.com which is an e-commerce website. The dataset set contained 6 categories with questions and their specifications.

For training, the dataset for category ‘mobile’ was selected. The total training data consisted of 740576 rows containing the columns, ‘qid’, ‘question’, ‘specification’. And the ‘target’. For the purpose of training, the records containing no answer were removed. Hence, the final training dataset consisted of 712427 records.

For the testing, the other categories which included AC, Backpack, Computer, Shoes, and Watches, was taken.

Architecture



Architecture for implementation of capsule network

Algorithm

Capsules communicate mainly through an iterative "routing-by-agreement" mechanism: a lower level capsule prefers to send its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule.
The lower level capsule will send its input to the higher-level capsule that 'agrees' with its input. This is the essence of the dynamic routing algorithm.

To explain the **dynamic routing algorithm**, let us take an example where a layer (l) contains **m** capsules and layer (l+1) contains **n** capsules.
For calculating the activation for jth capsule in layer(l+1) using activation of i^{th} capsule for layer(l), we first calculate the prediction vector for layer(l):

For a capsule j at layer l+1:

1. The prediction vector for a capsule j of layer (l+1) by a capsule i of layer(l) is given by:

$$\hat{U}_{ji} = W_{ij}\, u_{ij} \qquad (1)$$

Where W_{ij} is the weight matrix.

2. The output vector for capsule j is given as follows:

$$Sum_j = \Sigma c_{ij}\, \hat{U}_{ji} \qquad (2)$$

Where c_{ij} is the coupling coefficient.

3. The squash function is applied to the output vector, which is an activation function for the capsule layer. It helps in bringing the output vector in the range between 0 and 1.

The activation V_j for capsule j of layer(l+1) is given by:

$$V_j = \text{Squash}(\text{Sum}_j) \quad (3)$$

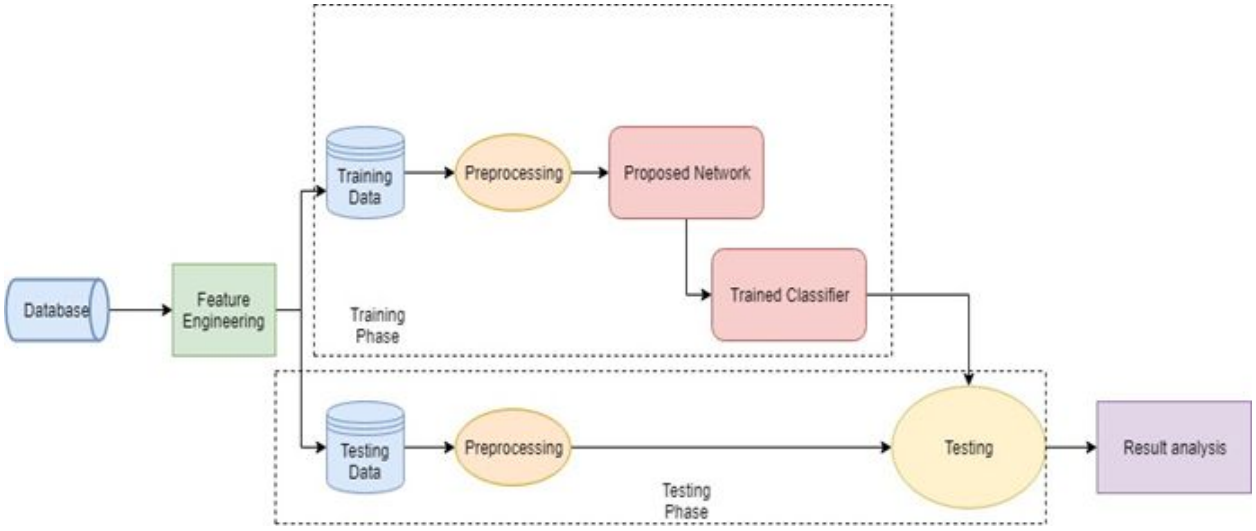
4. The last step is the routing agreement which is given by the following formula:

$$b_{ij} = b_{ij} + \hat{U}_{ji} V_j \quad (4)$$

Where b_{ij} depicts how strong the coupling is.

Now, if the prediction vector from lower level capsule, $\hat{U}_{ji}$, is in agreement with the output of higher level capsule, V_j, the term $\hat{U}_{ji} V_j$ wil increase which with in turn help in increasing b_{ij}**.** The large value of will depict the degree of correctness of capsule i for detecting capsule j.

METHODOLOGY



Proposed system

The training data was taken in various ratios such as 1:1 and 1:5 against each testing data. The training was done on the kaggle kernel GPU.

Preprocessing

The ‘question’ and ‘specification’ columns were merged together into a new column named ‘question_text’. For preprocessing the data, first, the short forms of words are changed to their original

forms such as changing ‘cause’ to ‘because’ or ‘i’m’ to ‘I am’. After correcting such spellings removal of the punctuations like{ ',', '.', '"', ':', ')', '(',} is carried out. Next, the numbers such as{1,2,3} and special characters such as {@, #, $}are removed along with stop words like {is, are, will}. All the missing values in the dataset are then replaced by a random string.

Embedding Matrix

For the embedding matrix, I used glove embedding and fasttext embedding. The final embedding matrix was calculated by taking the mean of the glove and fasttext embedding.

Bidirectional GRU

Bidirectional Gated Recurrent Unit or GRU has an internal mechanism which consists of gates, namely, input gate and the forget gate, to regulate the flow of information. GRU can be found in speech recognition models, speech synthesis and text generation. It is a combination of two independent Recurrent Neural Networks (RNNs) where one network is fed the input in normal time order and the other network is fed the input in reverse time order. At each timestamp, the concatenation of the two outputs is done.

For the Bidirectional GRU, the GRU length was taken as 256. For the activation parameter, ReLu activation function was used. The dropout rate was taken as 0.28.

Capsule Layer

Instead of focusing on the fixation points, The capsule layer helps the classifier to understand the context of the entire sentence.

For the Capsule layer, the number of routings were 3 and the number of capsules were taken as 10. The dimension of each capsule was set to 16.

TRAINING METHOD

Loss Function

A loss function or cost function is a function which is used to calculate the difference between the predicted output and the target, that is, it shows how well the network is performing. Here, Binary cross entropy is chosen with a logits loss function.

Optimizer

Here I have used Adam optimizer which is an adaptive optimization algorithm that has been custom made for training neural networks.

Training

The batch size of 412 was considered with the max_len parameter set to 70. The number of epochs were 30

PREDICTION METHOD

The softmax function gave the probabilities of each question-specification.

For the calculation of the number of hits, the probabilities for were then grouped into a list of lists on the basis of similar questions and were sorted in ascending order. Then, the target variables of the test dataset were compared to the probabilities at different indexes in the nested lists. Finally, the hits at various positions were calculated along with their probabilities.

Along with the hits, the **Mean Reciprocal Rank** and **confusion matrix** for each test dataset was calculated.

RESULTS

For the ratio 1:1, all the records for target=1 were selected and the same number of records for target=0 were selected. The results are formulated as follows:

Dataset	hit@1	HIT@2	hit@3	hit@5	MRR	Confusion matrix
AC	0.03	0.16	0.34	0.77	0.37	[[2052,1575], [26, 39]]
BKP	0.13	0.22	0.35	0.64	0.288	[[2548, 309], [144,21]]
COM	0.12	0.24	0.42	0.62	0.257	[[2127, 540], [32,18]]
SHO	0.21	0.4	0.56	0.77	0.41	[[645, 258], [68,27]]
WAT	0.12	0.28	0.42	0.79	0.307	[[925, 719], [23, 32]]

For the ratio 1:5, all the records for target=1 were selected and the number of records for target=0 were taken as five times the records for target=1. The results are formulated as follows:

Dataset	hit@1	HIT@2	hit@3	hit@5	MRR	Confusion matrix

AC	0.16	0.29	0.32	0.84	0.28	[[3494,133], [63, 2]]
BKP	0.21	0.39	0.54	0.85	0.42	[[2427, 121], [126,18]]
COM	0.18	0.32	0.44	0.64	.0.28	[[2625, 42]], [46, 4]]
SHO	0.25	0.37	0.52	0.77	0.41	[[854, 49], [81,14]]
WAT	0.14	0.28	0.43	0.70	0.33	[[1620, 24], [50, 5]]