CS-BOTS 1.0

Aditya, Akanksha Shrivastava

Guided by: Dr. Saurabh Bilgaiyan

School of Computer Engineering, Kalinga Institute of Industrial Technology,

1605004@kiit.ac.in, 1605006@kiit.ac.in

saurabh.bilgaiyanfcs@kiit.ac.in

Abstract- In this project, the authors have tried to implement the abstractive type of summarization using Sequence-to-Sequence RNN. Using this model, the output perceived is a summary which is short, lossy, and whose length does not necessarily depend on source text length and then the authors have used Bidirectional LSTM to find the best one to assign a particular genre. Both the models are made into RESTful APIs and then Client-Server Architecture (React.js-Python) is used to implore their end-user application.

Index Terms- Bidirectional LSTM, RESTful APIs, RNN, Sequence-to-Sequence models

I. Introduction

.In this day and age, as the internet gets increasingly cluttered for content, the comprehension of generated huge texts is growingly becoming a source of major inconvenience for a normal viewer and classifying it is another gruesome task. Text summarization and classification proves to be a boon as it abridges and categorizes the massive text into sizable length without removing the key information within it and assigning it to its genre [1]. The authors are proposing a model for unsupervised text classification and abstractive summarization of huge text using Sequence-to-Sequence RNN and Bidirectional LSTM. To implement this model and make a real world application, they are using React.js to interact with users taking various types of inputs. The frontend then sends the input to the backend to apply the pre-trained models and the resultant HTTP request is sent back to the UI to be displayed and visualized as Spider-graphs.

II. Study of similar projects or technology\ literature review

The authors are proposing a model for text classification and summarization of huge text using Sequence-to-Sequence RNN and Bidirectional LSTM. There is much research already done in this field but the task was to merge both the broad scenarios and provide results at one singular place. The paper also elucidates several explanations about Text Summarization. The work by Gupta, et al. is an overview in which they have clarified the facts about abstractive text summarization [1]. They have thoroughly demonstrated the broad categories of techniques of Text Summarization and have explained them in-depth [2, 3]. The paper also infers similar work done by Islam et al. in which they used Sequence-to-Sequence RNN to translate Bangla text [4]. They have furthered their study LSTM and have worked to implement this technique in the Python framework [4].

III. Basic concepts/ Technology used

Model for the Attention Mechanism in Sequence-to-sequence RNN (Summarization)

The sequence-to-sequence model consists of an encoder, a decoder, and an attention mechanism.

Encoder- It is bi-directional Long Short-term Memory (LSTM) layer that extracts information from the original text. It reads one word at a time and as it is an LSTM, it updates the hidden state based on the current word and the words it has read before [1]. In short, all an encoder does is it takes the input data, and passes the last state of its recurrent layer as an initial state to the decoder's first layer (recurrent).

Decoder- It is Unidirectional Long Short-term Memory (LSTM) that summaries one word at a time. The decoder LSTM starts working once it gets signaled that the entire source text has been read. It creates a probability distribution over the next word using information from the encoder and previously written text [1]. In short, the decoder takes the last state of the encoder's last recurrent layer and uses it as its first recurrent layer's initial state.

Attention Mechanism- Though this mechanism, the decoder can access the intermediate hidden states in the encoder and use all that information to decide which word is next [3].

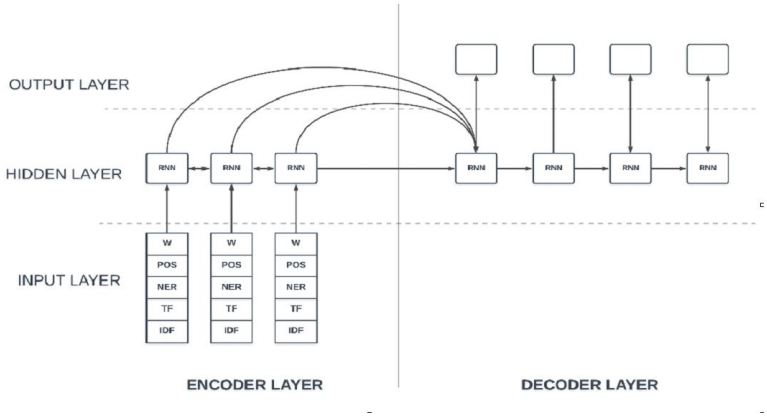


Fig 1.1 Attention Mechanism for Sequence-to-Sequence RNN

The algorithm is implemented in the following the steps:

1. The input text is encoded and the decoder is initialized with internal states of the encoder. The <start> token is passed as an input to the decoder.
2. The decoder is executed for a one-time step with the internal states.
3. The output will be the probability for the next word. The word with the maximum probability will be selected. The sampled word is passed as an input to the decoder in the next time step and the internal state is updated with the current time step.
4. The steps 3-5 are iterated until the <end> token is generated to the maximum length of the target sequence is hit.

Model for BiDirectional LSTM (Classifier Model) :

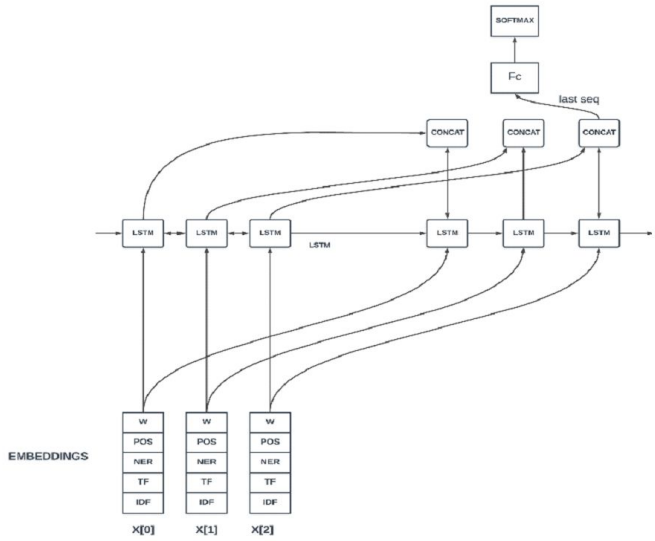


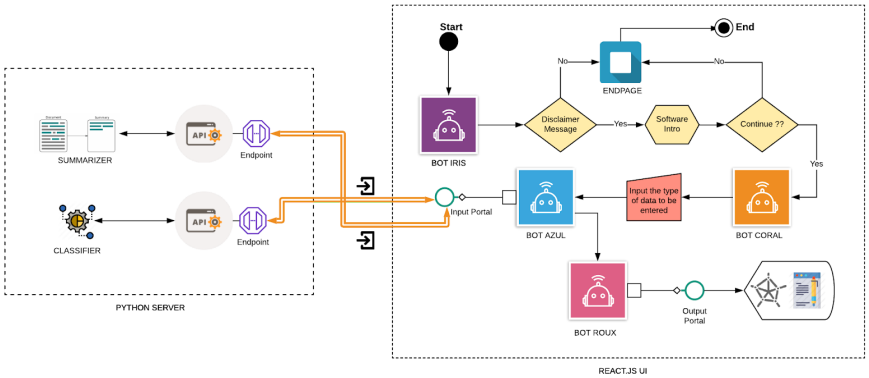
Fig 1.2 Bi-Directional LSTM

The algorithm is implemented in the following steps:

1. First the input layer is provided.
2. After adding the first embedding layer, a bidirectional LSTM layer is added.
3. The output layers are then added, i.e., Dense (activation = “ReLU”), Dropout, Dense (activation = “sigmoid”) in consecutive order.
4. The optimizer used to train the model is Adam’s optimizer and the loss parameter is ’binary_crossentropy’.

The work in this paper is largely inspired work from the paper of Nallapati et al. where they proposed the system to Attention Encoder-Decoder RNN, and achieve state-of-the-art performance and were able to address many critical problems in summarization [5]. The authors have tried to combine the objectives of the two and improvise and cater them onto a software so that users get to classify and summarize the text at a single place.

IV. Proposed Model / Architecture / Methodology/ Model Tool



The system works on step-by-step peer-call organisation, the UI-components work synchronously to work send state and props data transfer and the backend lays up RESTful APIs for summarization and

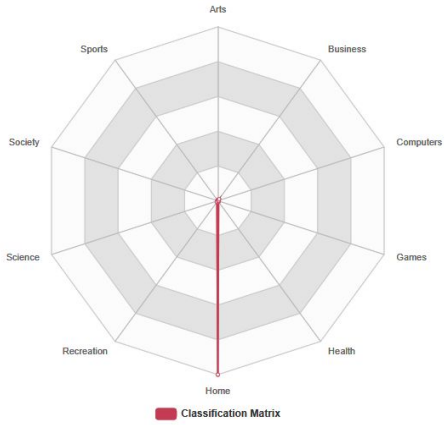
classification of textual data which on-call sends out the output summaries and classification matrix as value.

V. Implementation and Results

Implementation:

Frontend: The bots have been divided into Bots which handle the Components work-flow and send HTTP response to the python server.

Backend: The models have been previously trained and saved into pickle files which are then used to handle texts and then modelled into RESTful APIs to handle requests.



Results:

For text : best salad dressing delivered promptly quantities last vidalia onion dressing compares made oak hill farms sometimes find costco order front door want even orders cut shipping costs.

Summary: great product

Category: Home

VI. CONCLUSION

In this project, the authors have implemented a client-server architecture to apply Attention Mechanism on RNN to find the best possible summary. The model's output was then transferred to a bidirectional LSTM classifier model to assign a category. The result is visualised through Spider Graph. On the API part, the former model could be further improved by using beam-strategy for decoding the test sequences. Then the classifier model can also be improved by using Hierarchical Attention Networks or using CNNs and RNNs with a greater number of layers. On the other hand, the frontend could implement more options to provide the user a better experience (UX). The entire setup could be Dockerise for a close box application kind of feel.

References

[1] Gupta S., Gupta S. K., "Abstractive summarization- An overview of the state of the art", Expert Systems with Applications, Elsevier, Volume 121(1), 1 May 2019, Pages 49-65

[2] Gambhir M., Gupta V., "Recent automatic text summarization techniques: a survey", Artificial Intelligence Review, Springer, January 2017, Volume 47(1), pp 1–66

[3] Islam S., Mousumi S. S. S., Abujar S., Hossain S. A., "Sequence-to-sequence Bangla Sentence Generation with LSTM Recurrent Neural Networks", ICPC-AA - PerCAA 2019, Procedia Computer Science, Elsevier, Volume 152(1), 2019, Pages 51-58

[4] Nallapati R., Zhou B., Santos C., Gulçehre Ç., Xiang B., "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond", pp. 1-12