2020 IEEE International Conference for Innovation in Technology (INOCON)
Bengaluru, India. Nov 6-8, 2020

Abstractive Summarization of Meeting Conversations

Daksha Singhal[1], Kavya Khatter[2], Tejaswini A[3], Jayashree R[4]
[1,2,3,4]PES University, Bangalore, India
[1]daksha.singhal@gmail.com,[2]kavyakhatter1999@gmail.com,[3]tejaswini4299@gmail.com, [4]jayashree@pes.edu

***Abstract*—This paper contains the research work of encapsulating meetings by summarizing it which is within the style of dialogues. Throughout this process, the report will contain extractive information as an abstract of the text which is provided to the end-user. Abstractive and extractive methods are the foremost eminent techniques in text summarization. Extractive summarization is the most common summarization technique as it captures sentences based on either their frequency or using ranks associated with each of them and hence is unable to come up with grammatically correct sentences. The abstractive summarization may generate new vocabulary words apart from acquiring some extractive information from the text corpus. Here, we use the abstractive method of summarizing to train our model for dialogue systems.**

Keywords: Extractive summarization, Abstractive Summarization, deep learning models

I. INTRODUCTION

Nowadays data is expanding at a very fast rate and hence there is an imperative need for more capable and dynamic data summarizers. There has been an increasing amount of different ways for people to share and exchange information. Phone calls, e-mails, blogs and social networking applications are tools which have been in great use for communication. However, these are in the form of dialogues containing spontaneous utterances with speech disfluencies. This makes them cumbersome, complex and time-consuming to read.

Due to this rapidly increasing volume of data due to more data transfer through mobile traffic, accessing this the shortest possible time is an alarming issue these days.

It is a tedious task to gather all the data and give a summarized form. So, text summarizers came in hand that condense the document to a shorter version providing a clear and concise summary of the dialogue. A summary is beneficial as it helps in recouping long text files and thus saving time as well. This paper as a whole decodes the issues, challenges and how an abstractive method of summarization can be used for dialogue systems (i.e data involving conversations between two or more people).

As the paper involves abstractive summarization of dialogues, our data is in the form of dialogues involving a conversation between two people. The abstractive method includes the understanding of the text corpus followed by summarizing it while also maintaining semantic quality which is an important catalyst for improving the quality of the summary.

There are broadly VII sections in this paper consisting of introduction, literature survey, Implementation, Observation Results, Conclusion, Acknowledgement, References

II. LITERATURE SURVEY

A. Text summarization with pre-trained encoders:

The main proposal behind this model is to use transfer learning from a pre-trained BERT encoder and training the decoder from scratch. Edge of using this over LSTM is it helps in data parallelization and eliminating recurrence behaviour in Transformer models. Transformer based models generally create more logically and grammatically correct sentences.

B. A statistical approach to Anaphora Resolution

This paper consists of an algorithm which can be used for Pronoun resolution. Major factor taken into consideration was the distance between the pronoun and nearest antecedent. A probabilistic model is used for resolving pronouns.

C. Attention is all you need

This paper outlines an entirely different approach towards how the data can be trained parallel by using transformers instead of RNN's based models.

They also mention how multi-head attention and positional encoding helps in gaining good results and outperform most research on LSTM.

D. Summarization with Pointer-Generator Networks

This paper had an approach towards which words need to be replicated into generated summaries and therefore needs to be a different vocabulary.

There should be a probabilistic function which outputs a probability which acts as a decision boundary as there can be some vital information which should be in summary and not a different vocabulary for it.

III. IMPLEMENTATION

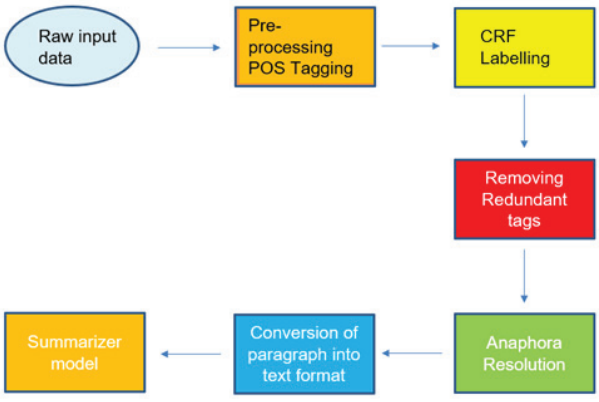

Fig. 1. Implememtation

978-1-7281-9744-9/20/$31.00 ©2020 IEEE

A. Dataset

The dialogue dataset that was used for the project is the **Switchboard Dialogue Act Corpus**.

It is a corpus of 1,155 5-minute telephone conversations between 2 persons, annotated with speech act tags. In these conversations, callers question receivers on provided topics, such as child care, recycling, and news media. 440 users participate in these 1,155 conversations, producing 221,616 utterances (we combine consecutive utterances by the same person into one utterance, so our corpus has 122,646 utterances).

B. Data preprocessing of Raw data

Unstructured data format files converted to structures CSV format for hassle-free preprocessing ahead.

C. Conditional Random Fields

CRF models the conditional probability and the decision boundary of different classes based on the context of a particular dialogue act and uttered sequence. All the DA labels will have different transition probability.

It is sometimes very difficult to guess the DA of dialogue without knowing what is the context of previous and followed dialogues.

For example, it's difficult to say if "Paris" is to be told as an answer or just a location, So for this, there should be an understanding of the last dialogue whether it was a question or not.

So we are using an approach in which the CRF tags depend on all the dialogues context assuming that the features are dependent on each other for learning to make predictions much better.

This type of approach takes the best of both HMM and MEMM.

The mathematical equation for CRF is given by:

$$p(\mathbf{y}|\mathbf{x}) = \underbrace{\frac{1}{Z(\mathbf{x})}}_{\text{Normalization}} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \underbrace{\theta_k}_{\text{Weight}} \underbrace{f_k(y_t, y_{t-1}, \mathbf{x}_t)}_{\text{Feature}} \right\}$$

This equation is comprised of broadly 2 terms:

Normalization: So when we expect that each tag should have a different conditional probability and when this prediction of tags takes place it gives us output in the form of probability, so in order to have all of them in one particular range, normalization is a must.

Weights and Features: weights are estimated using maximum likelihood function and features are defined and trained by our model.

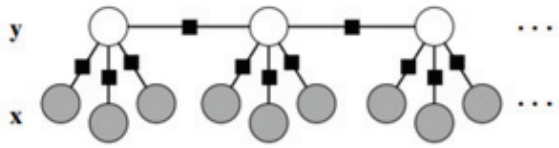

Fig. 2. HMM-based linear chain CRF

Fig. 3. CRF depending only on current observations.

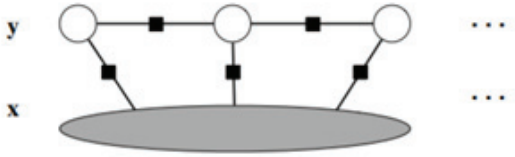

Fig. 4. CRF depending on all observations.

Building and Training a CRF Module in Python

For this classification problem, it's very important to notice a correlation between the utterance and the tags assigned to them.

For our approach, we have used python-pycrf module

D. Anaphora resolution

As we are dealing with dialogue format data, it's very important and necessary to know when there are references to other speakers in the dialogues.

In order to resolve this coreference problem, we have implemented anaphora resolution of our dialogue formatted data and resolved them into a text formatted one based on CRF labelled tags. This is considered as a pronoun resolution task which is a problem when we have more than 2 speakers.

To proceed with anaphora resolution, the following preprocessing steps were carried out.

- Removing brackets and its content
- Removing punctuations and symbols
- Expanding contractions
- Concatenating speaker name with its corresponding dialogue using CRF labelling

The quality totally depends on the spoken dialogue and context of the dialogue associated with each dialogue. For this, we have taken help from CRF tag labelling in order to get the context.

Our approach towards high-speed parsing was successful enough using spacy, Neuralcoref and NumPy and also included detections, features extraction and neural net computation.

Example of anaphora resolution:

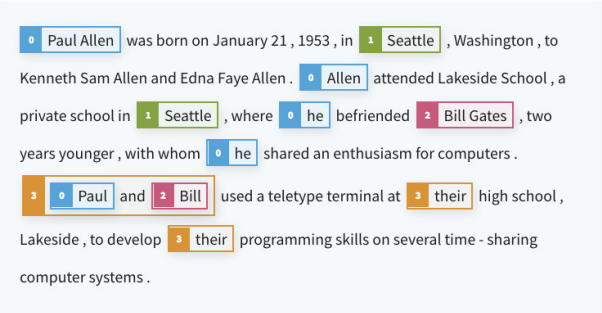

Fig. 5. Example of anaphora resolution

E. *Proposed Model*

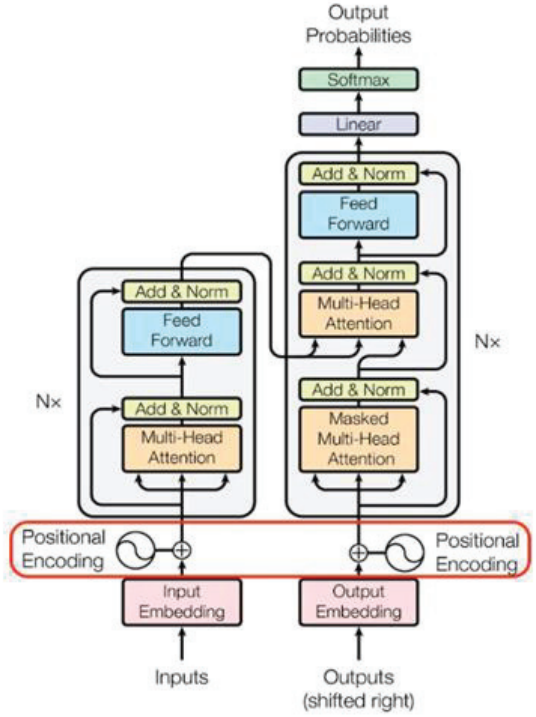

Fig. 6. Proposed Model

LSTM and RNN are more researched areas when it comes to text summarization but Transformers are surpassing them due to better performance. They significantly reduce training time and increase parallelization while also offering better results on machine translation. Transformers do not suffer from long dependency issues as they do not rely on past hidden states to capture dependencies with preceding words, they process a sentence as a whole and so there is no risk to lose (or 'forget') past information. Moreover, multi-headed attention and positional embeddings both provide information about the relationship and order of different words.

Transformer

The Transformer uses the encoder-decoder architecture with the integration of the multi-headed attention layer. The Transformer does the execution in a limited number of steps and in each step, It models all the words of a sentence nevertheless their position in the text by applying a self-attention mechanism.

Encoder and decoder both here are using fully connected layers.

F. *Model Architecture*

Encoder

Encoders are made up of N identical layers stacked upon each other.

In each encoder, there is a feed-forward network which has a position-wise fully connected layer and self-attention layer consisting of 8 heads running in parallel.

There is a residual connection around each of the two sub-layers which helps in avoiding the vanishing gradient problem in deep networks. It is followed by a normalization layer. The output of each sublayer is :

LayerNorm(x + Sublayer(x)).

Decoder

Decoders are made up of N identical layers stacked upon each other.

In each decoder, there is a feed-forward network which has a position-wise fully connected layer, self-attention layer consisting of 8 heads running in parallel and conducted over the output of the decoder stack. There is a residual connection around each of the two sub-layers which helps in avoiding the vanishing gradient problem in deep networks. For averting positions from attending to successive positions the multi-headed attention layer in the decoder stack is modified. It is followed by a normalization layer.

Attention

Multi-headed attention consists of four parts:

- Linear layers and split into heads.
- Scaled dot-product attention.
- Concatenation of heads.
- Final linear layer

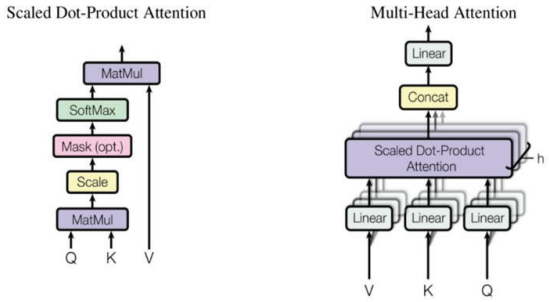

Fig. 7. Scaled Dot-Product Attention and Multi Head Attention

Mapping of the query, keys, values, and output is in all vector formats. The output is calculated by the weighted sum of the values. The weight that's assigned to each value is calculated by a compatibility function of the query with the corresponding key.

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T \sqrt{dk})V$$

The **Scaled Dot Product Attention** is applied to every head and in each attention step masking is used. The attention output for each head is then concatenated and put through a final Dense layer.

Instead of a single attention head, query vector, key vector, and value vector are divided into multiple heads that are created by multiplying embedding with the 3 matrices which are created during training. This allows the model to jointly attend to information at different positions from different representational spaces. The cost of computation of reduced dimensionality of each head after splitting is the same as a single head's attention with full dimensionality.

Setup

While training our model we experimented with a number of hyperparameters. Adam was the optimizer used with a custom learning rate schedule. This denotes that the learning rate increases linearly for the first warmup_steps training steps, and decreases thereafter the maximum epochs at 400 with early- stop strategy. Hidden vectors' size is set to 512. We trained it on 750 examples and validated on 40 examples. The loss metric used here is a cross-entropy loss for predicting one word at a time at the decoder and appending it to the output.

IV. Observation and Results

Rouge scores were used as the standard metric for the evaluation and inference of the summarizer. It works by comparing an automatically produced summary or translation against a set of human-generated reference summaries.

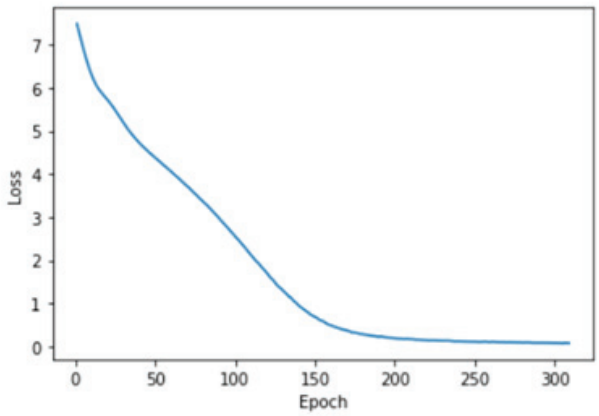

Fig. 8. LOSS vs EPOCHS

```
Evaluation with Avg
        rouge-1:        P:  0.35        R:  0.46        F1:  0.40
        rouge-2:        P:  0.14        R:  0.18        F1:  0.16
        rouge-3:        P:  0.08        R:  0.11        F1:  0.10
        rouge-4:        P:  0.07        R:  0.09        F1:  0.08
        rouge-l:        P:  0.29        R:  0.36        F1:  0.32
        rouge-w:        P:  0.17        R:  0.09        F1:  0.12

Evaluation with Best
        rouge-1:        P:  0.35        R:  0.46        F1:  0.40
        rouge-2:        P:  0.14        R:  0.18        F1:  0.16
        rouge-3:        P:  0.08        R:  0.11        F1:  0.10
        rouge-4:        P:  0.07        R:  0.09        F1:  0.08
        rouge-l:        P:  0.29        R:  0.36        F1:  0.32
        rouge-w:        P:  0.17        R:  0.09        F1:  0.12
```

Fig. 9. ROUGE SCORES

TABLE I.

Model	Rouge Score
LSTM-RNN	0.21
Transformer+multi head Attention	0.46

Using LSTM-RNN's summarized text had a lot of repeated words due to which it has a lesser rouge score as compared to the transformer model.

Using the Transformer model the summarized text had relatively less new vocabulary getting generated due to the smaller amount of training data.

V. Conclusion

We presented supervised abstractive summarization on Switchboard Dataset using a transformer model that reduces n-gram blocking to reduce repetition and successfully summarized the dialogues. Future works include the model training on a transformer with a pointer generator and additional hyper-parameter tuning for a state-of-the-art summarizer. Training and analyzing our model on different dialogues like the Google Dialogue Dataset.

Acknowledgement

We would like to thank our mentor and guide Dr Jaishree.R for the support and guidance over the course of the project. Finally, we would like to thank the Computer Science Department of PES University for providing the opportunity to work on this project.

References

[1] Attention is all you need: Yu-Hsiang Huang.

[2] Prakhar Ganesh and Saket Dingliwal. 2019. Abstractive summarization of spoken and written conversation.

[3] "Gettothepoint: Summarization With Pointer-generator networks"

[4] Just News It: Abstractive Text Summarization with a Pointer-Generator Transformer Vrinda Vasavada, Alexandre Bucquet

[5] Speaker-change Aware CRF for Dialogue Act Classification Speaker-change Aware CRF for Dialogue Act ClassificationGuokan Shang1,2, Antoine J.-P. Tixier1, Michalis Vazirgiannis1,3, Jean-Pierre Lorr

[6] Text Summarization with Pre-trained Encoder: Yang Liu and Mirella Lapata

[7] Automatic Dialogue Summary Generation for Customer Service: Chunyi Liu, Peng Wang, Jiang Xu, Zang Li, Jieping Ye

[8] Automatic Chinese Dialogue Text Summarization Based On LSA and Segmentation: Chuanhan Liu, Yongcheng Wang, Fei Zheng, and Derong Liu

[9] Summarizing Online Conversations: A Machine Learning Approach: Arpit Sood, mohamed.thanvir@students.iiit.ac.in, Vasudeva Varma