Speech To Text Summarization

Karthika Kamath, Mahathi Parvatham, Rashmi Nagpal, Shikhar Seth, Shubham Goel

Plaksha TechLeaders Fellowship, Gurugram, India
Submitted 11th January 2020

Abstract

As a core task of natural language processing and information retrieval, automatic text summarization is widely used in many places. This paper describes a novel hybrid summarization model, which generates text from audio input using two independent components : an automatic speech recognizer and a summarizer. To verify the performance of our model, we compared it with the current popular automatic text summary model on CNN/Daily Mail dataset, and then used ROUGE(Recall-Oriented Understudy for Gisting Evaluation) as well as BLEU score metrics as our evaluation method.

Keywords: Automatic Speech Recognition, Text Summarization, Summary Evaluation

Introduction

The internet age has brought unfathomably large amounts of data to the fingerprints of billions - if we only had time to read through it. Though our lives have been transformed by ready access to limitless data, we also find ourselves ensnared by the information overload. For this reason, automatic text summarization - the task of automatically condensing a piece of text to a shorter version is a need of hour.

Summarization is a tough problem because the system is unable to understand the context of text since it requires semantic analysis as well as coherency within similar contextual data. Therefore, attempts at performing true abstraction have not been successful so far while there are broadly two types of summarization paradigms: extractive as well as abstractive.

The extractive method extracts the important sentences or a section of text from the original text and then combines to form a summary, while the abstractive model generates novel words that do not exist in the source text and combines them to form a summary by retaining the original meaning. Compared with the difference between them, the extraction paradigm is relatively simple and ensures grammatical correctness, but the semantics are inconsistent; while the abstraction paradigm is more concise, but redundant. When summarizing a very long text, the extractive approach is too simple and the readability is poor, and the abstract method of compressing a long input sequence with a single fixed-length vector may cause the information loss, neither of them could perform the long text summary better. At present, some neural network models combine the advantages of extractive and abstractive approaches i.e. select the key sentences from the source text using extractive methodology and then generate the summary of those sentences by using abstract methods.

Inspired by [1], we propose a novel method which integrated the extractive as well as abstractive method on CNN/Daily Mail dataset. Since each word contributes differently to the sentence semantics, we used ROUGE (Recall-Oriented Understudy for Gisting Evaluation) as well as BLEU score for our evaluation metrics.

The rest of the paper is organized as follows. Section II comprises related work. In Section III the proposed approach is presented in detail. Further, we showed the results and analysis in Section IV. Section V comprises a conclusion as well as future work.

Related Work

Due to the difficulty of abstractive summarization, the great majority of past work has been extractive.

Denil et al. [2014][2] use a hierarchical ConvNet architecture (CNN) divided into a sentence level and a document level. The sentence level learns to represent sentences using their words and the document level learns to represent the input document using the first level. Then these representations are used to score how pertinent a sentence can be towards the entire document.

In [Cao et al., 2015][3], Recursive Neural Networks (R2N2) is used to rank sentences for multi-document summarization. They use two types of hand-crafted features as inputs: word features and sentence features. This enables the system to learn how to score sentences based on a hierarchical regression process which uses a parsing tree to score sentence constituents such as phrases.

Kavita Ganesan et al. [4] proposed a novel graph-based summarization framework (Opinosis) that generates compact abstractive summaries of extremely redundant opinions. It has some distinctive properties that are crucial in generating abstractive summaries: Redundancy Capture, Gapped subsequence capture, Collapsible structures. The model generates an abstractive summary by exhaustively searching the Opinosis graph for appropriate sub-graphs encoding a valid sentence and high redundancy scores. The major components of the system are meaningful sentence and path scoring. Then a valid path is selected and it's marked with high redundancy score, collapsed paths and generation of
summary. Then all the paths are ranked in the descending order of the scores and are eliminated duplicated paths by
using Jaccard measure.

The recent success of sequence-to-sequence models (Sutskever et al.) [5] in which recurrent neural networks (RNNs) both read and freely generate text, has made abstractive summarization viable. Alternatively, reinforcement learning (RL) can give room for exploration in the search space. . This approach makes an end-to end trainable stochastic computation graph, encouraging the model to select sentences with high
ROUGE scores. However, they define a reward for an action (sentence selection) as a sentence level ROUGE score between the chosen sentence and a sentence in the ground truth summary for that time step. This leads the extractor agent to a suboptimal policy; the set of sentences matching individually with each sentence in a ground truth summary isn't necessarily optimal in terms of summary-level ROUGE score.

Though these systems are promising, they exhibit undesirable behavior such as inaccurately reproducing factual details, an inability to deal with out-of-vocabulary (OOV) words, and repeating themselves.

Methodology

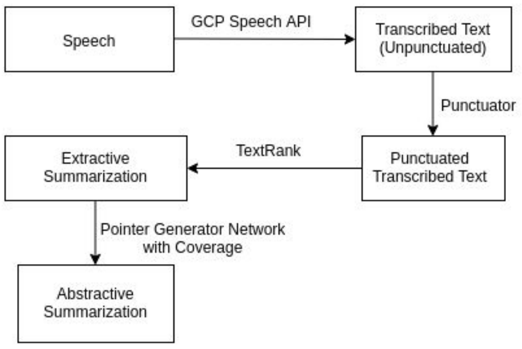

Speech
GCP Speech API
Transcribed Text (Unpunctuated)
Punctuator
Punctuated Transcribed Text
TextRank
Extractive Summarization
Pointer Generator Network with Coverage
Abstractive Summarization

Figure 1. Automatic Speech Summarization System with text presentation

The figure describes the pipeline followed for executing the task of speech to text summarization. Through the interface a speech file in wav format is uploaded. If it is any other format it is converted to wav for further processing using pydub library. This speech file is converted into a text file using google speech recognition library resulting in a transcribed text without any default punctuation so we have to punctuate it using the punctator library to be processed by the TextRank library that generates the extractive summarization. We used a hybrid model of extractive text summarization in order to select highest score sentences from the input text data that is used to abstractedly summarize in order to produce better results. The abstractive summarization has a pointer generator network as its underlying model. That summarized text is finally displayed on the user interface.

Speech to Text

Attention based encoder-decoder architecture such as Listen, Attend and Spell (LAS) subsume acoustic, pronunciation and language model components of a traditional automatic speech recognition system into a single neural network[8]. In our work, we used Google SpeechToText API for speech recognition and transcription. That very model integrates acoustic, pronunciation and language models into a single neural network, and does not require a lexicon or a separate text normalization component. On top of that, we used Punctuator pre-trained model to retain the syntactic structure of data.

Text Summarization

Text Summarization is broadly divided into two categories : Extractive & Abstractive. For the former, methods rely on extracting several parts such as phrases and sentences, from a piece of text and stack them together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method.

Abstractive text summarization aims to shorten long text documents into a human readable form that contains the most important facts from the original document. Sometimes, words in the summarized version don't even occur in the original text.

Extractive Summarization

I) TextRank

It is a graph based ranking model for text processing. It's basic idea is that of "voting" or "recommendation" [7]. When one vertex links to another vertex, it basically casts votes for that vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex, casting the vote determines how important the vote itself is, and that information is used for ranking the model. For extractive summarization, we used TextRank algorithm, to extract sentences which rank the entire sentences, and adds a vertex to the graph for each sentence in that text. The resulting graph which is produced on using this algorithm, is highly connected with a weight associated with each edge, indicating the strength of the connections which is established between various sentence pairs in the text and the text is therefore, a weighted graph.

Abstractive Summarization

I) Seq2Seq using Attention and Beam Search

A sequence to sequence model is one in which a series of words is fed as input that is processed one after another to output another series of words. The model is composed of an encoder and a decoder. Both encoder and decoder are recurrent neural networks. The output state of the encoder called the context vector which is an array of numbers is given as input to the decoder. In the case of text summarization given a corpus of sentences to be summarised to the model it should output the summary in given number of words.

The architecture used is as follows:

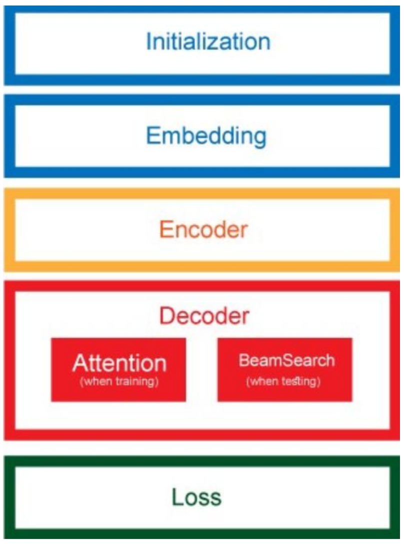
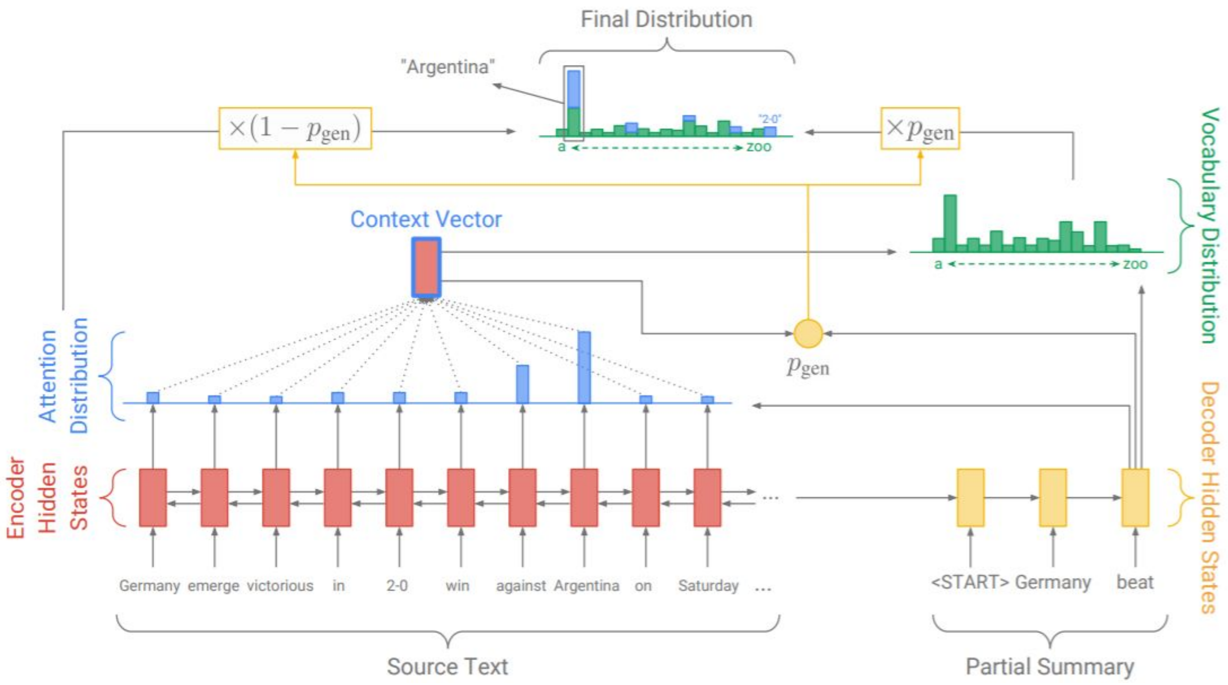


Figure 2. Sequence-to-Sequence Model Architecture

In the initial block tensorflow placeholders and variables are initialised along with building the RNN model architecture class that would be used for encoder and decoder. In the embedding block we need to preprocess the dataset so as to feed into the next module. Further, Glove(Global Vectors for Word Representation) was used to create word embeddings. In the next block, an encoder which is the multilayer bidirectional long short term memory network which comprises of forward and backward cells and stacked bidirectional dynamic recurrent neural network are used to connect them together. This will give two main outputs, encoder output and encoder state. Encoder output would be used for attention calculation and encoder state returns the initial state of the decoder. Then comes the decoder block which has two parts. For the training part it uses attention and for testing it uses beam search. Attention allows models to focus on relevant parts of the sentences and is useful when the sentence is very long. Beam search is an approximate search strategy to choose better results from all possible candidates. For this beam width needs to be specified, if beam width is one then it is the same as greedy search. Final block is the loss block. This block is where training actually occurs where the multiple steps are calculating loss, calculating gradient and applying clipping on the gradients and applying optimizers. We will use sparse softmax cross entropy loss and adam optimizer. Even though the results given were acceptable, it faced few issues like out of vocabulary word and factual information generation error. Out of vocabulary words problem is faced because the model is trained on a limited vocabulary but while testing if a new word is not recognized by the model then it assigns an unknown tag : <unk>. And the latter issue arises because the factual information are unique tokens which are supposed to be copied as it is and not supposed to be generated by the model.

II) Pointer Generator Networks

The problem encountered in the sequence-to-sequence model can be overcome by using a pointer in the model. The factual errors occur due to the rate of Out-of-Vocabulary (OOV) words. These errors generated in that model don't occur in this model as words can be copied by pointing to words in the source text or by words that are generated from a fixed vocabulary. The basic structure is similar to the sequence-to-sequence model with attention but with an additional feature to it.



In addition to getting the attention distribution and vocabulary distribution, we also include a probability distribution. It is basically used as a switch to help the model choose whether to copy the word from the input sequence or pick a word from the vocabulary dictionary. The generation probability p_{gen} for each timestep is computed using the context vector, the decoder state, and decoder input. The context vector is a weighted sum of the attention weights while the decoder state and input are parts of the RNN used.

By using the probability distribution, words which were previously unknown and represented by *'UNK'* token can now be replaced by the actual words as even though it is not in the dictionary, it can be simply copied from the input source. Also, the numbers which were incorrectly substituted before are now able to be reproduced correctly.

Another problem encountered in the sequence-to-sequence approach is - repetition. This occurs due to the attention vector attending the same word repeatedly. To prevent this, a concept called coverage can be used. Coverage is the cumulative attention which is nothing but whatever has been covered so far. It is used as an additional input to the attention mechanism. It will penalize words that have already been used before in the summary.

This approach is a very optimal solution as it involves both fine-grained extraction while copying the words (extractive) while abstractedly generating.

$$P_{final}(w) = p_{gen} * p_{vocab}(w) + (1-p_{gen}) \sum_{i:wi=w} a_i$$

Results

The abstractive summarization was tested on 25 samples of the Inshorts Dataset. The scores below are the average of each article's score.

BLEU SCORE
0.704

Table 1. AVERAGE BLUE SCORE

ROUGE 1	ROUGE 2	ROUGE L
0.329	0.134	0.251

Table 2. ROUGE SCORE (Avg F1 Score)

Evaluation Metrics

The evaluation of a summary quality is a very ambitious task. Serious questions remain concerning the appropriate methods and types of evaluation. There are a variety of possible bases for the comparison of summarization system performance. We can compare a system summary to the source text, to a human generated to another system generated summary. In our work, we used existing metrics i.e. BLEU and ROUGE to match the system summary against the “ideal summary”, though ideal summary is hard to define.

I) BLEU- Bilingual Evaluation Understudy

It is one of the first metrics which was close to the human judgments of quality. This was originally developed for Machine Translation but can be used for evaluating a generated sentence to a reference sentence. A perfect match indicates a score of 1.0 and mismatch 0.0. This approach works by counting the matching n-grams in a generated sentence to a reference sentence. Unigrams, bigrams, trigrams, and 4-grams are made up of chunks of one, two, three and four words respectively. Although it is easy to calculate, it doesn't consider meanings or sentence structure.

II) ROUGE- Recall-Oriented Understudy for Gisting Evaluation

It is an evaluation metric used for automatic summarization as well as machine translation. Recall in this approach refers to how much of the summary is actually recovering the actual content. Precision refers to how much of the words in the system are recovered and can be defined as the number of overlapping words by the total words in the summary system. It gives us bad results when there are quite a few extra words apart from the actual text. This implies that its words only for concise summaries generated. That is why it is better to compute the Recall and then report the F-Measure.

There are different kinds of ROUGE metrics also, like:

- ROUGE-N - it takes into account the various n-gram overlap
- ROUGE-S - Is any pair of words in the sentence that is in the same sequence. This may include random gaps in the sequence of words
- ROUGE-L - It is the longest matching sequence of words. It is the longest in-order common n-grams

For summarization, automatic metrics such as ROUGE and BLEU have serious limitations:

- They only assess content selection and do not account for other quality aspects, such as fluency, grammaticality, coherence, etc.
- To assess content selection, they rely mostly on lexical overlap, although an abstractive summary could express they same content as

a reference without any lexical overlap.

- Given the subjectiveness of summarization and the correspondingly low agreement between annotators, the metrics were designed to be used with multiple reference summaries per input. However, recent datasets such as CNN/DailyMail and Gigaword provide only a single reference.

Therefore, tracking progress and claiming state-of-the-art based only on these metrics is questionable. Most papers carry out additional manual comparisons of alternative summaries. Unfortunately, such experiments are difficult to compare across papers.

Codebase and Presentation

We have uploaded our codebase and presentation as well as the report at below mentioned github repository. https://github.com/KK945/DataX-2-PlakshaTLF . This repository has been maintained to show the transparency of our models as well as dataset used.

Conclusion

Implemented an end to end pipeline for speech to text summarisation using various modules. Speech to text conversion was carried out followed by text preprocessing and punctuation that was fed to the summarisation module. Used a hybrid approach of extractive summarisation followed by abstractive summarisation for producing the summary. Evaluation metric used were ROUGE and BLEU.

References

[1] Chen, Y.C.; Bansal, M. Fast abstractive summarization with reinforce-selected sentence rewriting. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 675–686.

[2] Denin, Demiraj, Blunsom, Freitas N. Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network. Arxiv

[3] Cao, Wei, Dong, Li, Zhou. Ranking with recursive neural networks and its application to multi-document summarization. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence; January 2015; Pages 2153–2159.

[4] Ganesan, Zhai C, Han. Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010); pages 340–348.

[5] Sutskever, Vinyals, Quoc V Le. Sequence to Sequence Learning
with Neural Networks. Advances in neural information processing systems, 2014

[6] Bansal, Chen. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 675–686.

[7] Mihalcea, Rada & Rada, & Tarau, Paul. (2004). TextRank: Bringing Order into Texts.

[8] Chiu, Chung-Cheng & Sainath, Tara & Wu, Yonghui & Prabhavalkar, Rohit & Nguyen, Patrick & Chen, Zhifeng & Kannan, Anjuli & Weiss, Ron & Rao, Kanishka & Gonina, Ekaterina & Jaitly, Navdeep & Li, Bo & Chorowski, Jan & Bacchiani, Michiel. (2018). State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. 4774-4778. 10.1109/ICASSP.2018.8462105.