CS230

An Examination and Application of Text Summarizer (Textsum) on Amazon Customer Reviews

YI HE
Stanford Center
for Professional Development
yihe1008@stanford.edu

CAN JIN
Stanford Center
for Professional Development
cjh0511@stanford.edu

Abstract

Text Summarization, the process of creating a short and logical version of a longer document, is a complex problem which, in spite of the progress in the area thus far, poses many challenges to the scientific community. For this project, we focused on improving the existing text summarizer, Google's Textsum [1]. We introduced Universal Sentence Encoder and ConceptNet NumberBatch to improve the current Textsum model. Additionally, we explored, transfer learning, whether a text summarizer trained on CNN/DailyMail can be used to accurately summarize Amazon reviews. With the newly improved model, we applied it to Amazon book reviews to qualitative test the accuracy.

1 Introduction

Textual information in the form of digital documents quickly accumulates to huge amounts of data. Most of this large volume of documents is unstructured: it is unrestricted and has not been organized into traditional databases. Processing documents is, therefore, a perfunctory task, mostly due to the lack of standards [2]. There is a great need for automatic text summarization since we neither can create summaries of all of the text manually nor read all the text.

Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning [3]. It has proven to be difficult since the problem involves a good understanding of a language model to reproduce both structurally correct and meaningful sentences. Given those challenge tasks, we focused on improving the existing text summarizer, Textsum, in this project. Our model is trained on CNN/DailyMail dataset. The input to our algorithm is the first two sentences of each article and the first bulleted point as the gold label sentence (target variable). The output of our model is a concise summary of the article.

2 Related work

There are two main approaches to summarizing text: Extractive methods and Abstractive methods. Extractive text summarization involves the selection of phrases and sentences from the source document to make up the new summary. While Abstractive text summarization involves generating entirely new phrases and sentences to capture the meaning of the source document. This is a more challenging approach but is also the approach ultimately used by humans.
Classically, most successful text summarization methods are extractive because it is an easier approach, but recently deep learning methods have shown promising results for text summarization as the abstractive approach. The results of

deep learning methods are not yet state-of-the-art compared to extractive methods, yet impressive results have been achieved on constrained problems such as generating headlines for news articles that rival or out-perform other abstractive methods.

We used Textsum as our baseline. Textsum uses a deep learning method by developing the problem of text summarization as a sequence-to-sequence learning problem.

3 Dataset and Features

Google's Textsum was developed using the Gigaword dataset, not open sourced and inaccessible by Stanford students. However, recent research on summarizer has moved toward using the CNN/DailyMail dataset [4]. This dataset has the original text and a short set of summarized bullet points that represent meaningful "highlights" of each article.

The CNN dataset contains 92,570 articles, while the DailyMail contains 219,503 articles. Moreover, each article includes 3 to 4 bullet points. We use the first two sentences of each article as model input, and the first bullet point as the gold label sentence. We split the dataset to train/dev/test as 80:15:5. Below is an example from CNN/DailyMail.

> ***First two sentences****: (CNN)Island-hoppers take note: Providenciales in the Turks and Caicos is TripAdvisor's latest pick for the world's top island. Providenciales climbed a rung from last year's No. 2 ranking among the travel review site's Travelers' Choice award winners to bump Ambergris Caye out of the top spot.*
>
> ***Gold label sentence****: Providenciales in the Turks and Caicos is the top TripAdvisor Travelers' Choice award winner for islands.*

We also processed each article using the Sent_Tokenize from the NLTK package. We limited the vocabulary size to 200k. We created a vocabulary list which outputs word count frequency. Also, we converted each article to a sequence of a serialized Tensorflow object.

With the newly improved model we trained on the CNN/DailyMail dataset, we then applied it to Amazon book reviews [5].

4 Methods

4.1 Abstractive Summarization Architecture

We used an open source Tensorflow model, Textsum [1], as our baseline. Textsum uses an encoder-decoder model with a bidirectional LSTM-RNN encoder and an attentive unidirectional LSTM-RNN decoder with beam search. The encoder-decoder model is trained end-to-end [6]. We followed the approach described by Danqi C. (2017) to train our baseline model. For the first decode timestep, we fed in the last output of the encoder as well as the embedding for the start (<s>) token. For subsequent decode timesteps, the decoder uses the last decoded output in addition to the word embedding for the previous word to generate the next word. During the train step, the previous word is the actual previous word from the gold label sentence, but during the decode step, the previous word is the previously-generated word. The decoding process will continue until the generated summary reaches the max decode length set at 30, or until it generates an EOS token [6]. Figure 1 illustrates the Textsum architecture.

4.2 ConceptNet NumberBatch

Google trained Textsum on Gigaword dataset, which contains 10 million article-summary pairs. Because of the limited amount of training data in CNN/DailyMail dataset, we initialized our word embeddings with ConceptNet Numberbatch to give our model a more powerful semantic representations of the source input tokens.

Conceptnet Numberbatch is a set of semantic vectors: it associates words and phrases in a variety of languages with lists of 300 numbers, representing the gist of what they mean. ConceptNet is collected from a combination of expert-created resources, crowdsourcing, and games with a purpose.

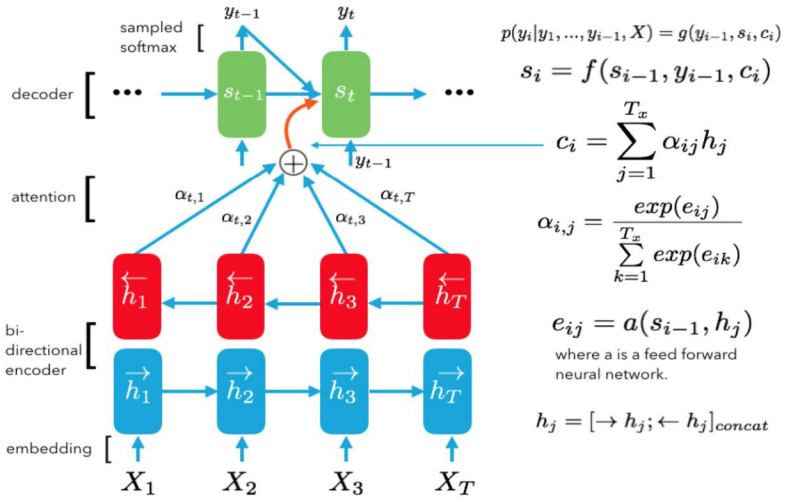

Figure 1: Attentive encoder-decoder model, where each hidden unit represents an LSTM cell [6]

4.3 Universal Sentence Encoder

As previously mentioned, we initially used the first two sentences of each article as our model input. However, we found that the main idea is not properly captured in the first two sentences for some articles. Instead of using the first two sentences, we applied Universal Sentence Encoder to find the two most similar sentences to the gold label sentence then fed them into our model. Take the below example:

> ***First two sentences***: *(CNN)The "Star Wars" universe keeps on expanding. Details of the plot for 2016's "Star Wars: Rogue One" were revealed during a panel at Star Wars Celebration fan festival Sunday in Anaheim, California.*
>
> ***Gold Label Sentence***: *The plot focuses on a rogue mission to steal plans for Death Star.*

Universal Sentence Encoder is a pre-trained model for encoding sentences into embedding vectors that specifically target transfer learning to other NLP tasks

4.3 Evaluation

ROUGE is a common metric for automated text summarization tasks that computes a number of co-occurrence statistics between a predicted output and true output [6]. We run ROUGE on our generated summaries and gold label for the test set.

5 Experiments/Results/Discussion

5.1 Baseline

We used Textsum as our baseline. Similar to the results from Danqi C. (2017), the results we produced from our baseline model is inadequate on the CNN/DailyMail dataset. After training on 80% of the data set, we noticed a large number of <UNK> tokens in the summaries that made it unreadable. After investigating on Github, we found out that we ran into the same issue as other researchers who are trying to use the same model and dataset. CNN/DailyMail contains 312k

articles, which is much fewer than Gigaword's 10 million articles used by Textsum. The discrepancy between the number of articles between the two data sets resulted in such different outcomes.

Two problems we identified in the existing models are a lack of generalization in using longer articles, and an inability to use the source text itself to provide words and relying on using <UNK> tokens instead [6]. The example below illustrates the problem in our initial output.

> **First two sentences**: *(CNN)Island-hoppers take note: Providenciales in the Turks and Caicos is TripAdvisor's latest pick for the world's top island. Providenciales climbed a rung from last year's No. 2 ranking among the travel review site's Travelers' Choice award winners to bump Ambergris Caye out of the top spot.*
>
> **Gold Label Sentence**: *Providenciales in the Turks and Caicos is the top TripAdvisor Travelers' Choice award winner for islands.*
>
> ***Decoded output**: Providenciales is <UNK><UNK><UNK> TripAdvisor <UNK>.*

5.2 ConceptNet NumberBatch

As mentioned above, because of the limited amount of training data in CNN/DailyMail dataset, we initialized our word embeddings with ConceptNet Numberbatch to give our model a more powerful semantic representations of the source input tokens. With this approach, we found the decoded results had significantly less <UNK> tokens. The decode output for the article above improved a lot.

> ***Decoded output**: Providenciales is the top TripAdvisor <UNK> winner for top island.*

5.3 Universal Sentence Encoder

Instead of using the first two sentences, we applied Universal Sentence Encoder to find the two most similar sentences to the gold label sentence then fed them into our model. We obtained better decoded summaries and slightly better Rouge score with this approach.

5.4 Training

Our model was trained for more than 10 epochs. While conducting our hyperparameter search, we primarily experimented with the learning rate, batch size, number of encoding layer, number of hidden units, number of softmax samples. Throughout the training, Rouge score was our primary evaluation metrics. Since using 1 GPU takes about 1 week to train the model, we used a subsample of the training data to tune the parameters. The parameters in our final model are – learning rate =0.15, batch size = 4, encoding layer = 4, number of hidden units = 256, number of softmax samples = 4,096, word embedding size = 300. Figure 3 illustrates the running average loss in our final model.

5.5 Results

	Preprocessing	Learning model	Rouge-1	Rouge-2	Rouge-L
1	None	Textsum	0.078	0.004	0.0725
2	Tokenized	Textsum + CN	0.108	0.019	0.102
3	**Tokenized**	**Textsum + CN + USE**	**0.122**	**0.036**	**0.110**

Table 1: Rouge score for our three experiments

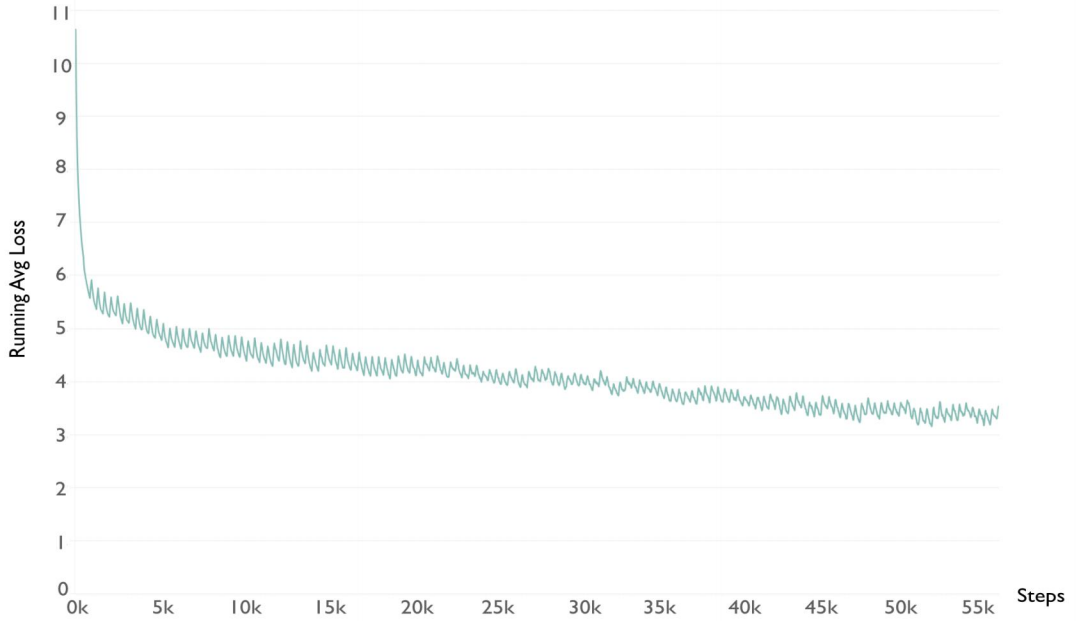

Figure 3: Running average loss of the final best model

Textsum + CN + USE gives us the best rouge scores on the CNN/DailyMail data set. The Rouge-1 score improved significantly comparing to our baseline model – Textsum.

5.6 Application on Amazon Book Reviews

Once we applied the model on Amazon book reviews, we got unfavorable decoded results. Since the diction between news articles and Amazon reviews are vastly different, the decoded results followed the word usage in news for Amazon reviews. However, we were surprised to find that the model showed good sentimental understanding: on a positive Amazon review the decoded message showed a positive news piece while on a negative Amazon review the decoded message showed a negative news piece. Below is an example that shows this surprising result:

> ***Amazon Book Review****: "This is a great book. It has twists and turns and many surprises. One of those stories I would read again. There is so much I am sure I missed something."*
>
> ***Decoded output****: I made for great payment, simply for our <UNK>.*

6 Conclusion/Future Work

In conclusion, the original TextSum model did not work well on small data sets. However, once we introduced Universal Sentence Encoding and ConceptNet NumberBatch, we were able to get better Rouge scores. We believe that by providing the model with similar sentences to the main idea and an improved embedding layer, the model can perform better with smaller data sets. However, transfer learning, the hypothesis that training with news articles can summarize book reviews, did not perform as well as we had hoped.

There are certain areas for improvements on this project, and if we had more time on this project we would like to execute them. First, add Amazon reviews and other non-news data sources to the training dataset. Second, instead of using only two sentences for training, we will use more sentences from each article to train the model. Finally, the model takes about a week to train on 1 GPU, we would like to purchase more GPUs and computing time to fine-tune the parameters of the model.

7 Contribution

We have two team members and we worked equally on the project. Yi worked on the training infrastructure and model architecture development. Can contributed to model tuning, result analysis and visualization.

Our code is on GitHub at *https://github.com/yihe1008/yi_can_summarize*.

References

[1] Chen, D., Bolton, J., & Manning, C. D. (2016). A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task. Acl 2016, 2358–2367.

[2] Torres-Moreno, J. (2014). Automatic Text Summarization (Cognitive Science and Knowledge Management).

[3] Mehdi, A., Seyedamin, P., Mehdi, A., Saeid, S., Elizabeth, D., Juan, B., & Krys, K. (2017). Text Summarization Techniques: A Brief Survey.

[4] Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. <*https://cs.nyu.edu/~kcho/DMQA/*>

[5] McAuley, J., & Leskovec, J. (2013), Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys.

[6] Vincent C., Eduardo M., Liezl P., & Danqi C. (2017). An Examination of the CNN/DailyMail Neural Summarization Task.

[7] Cer, D., Yang Y., Kong S., Hua N., Limtiaco N., St.join R., Constant N., Guajardo-Cespedes M., Yuan S., Tar C., Sung Y., Strope B., & Kurzweil R. (2018). Universal Sentence Encoder.

[8] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation By Jointly Learning To Align and Translate. Iclr 2015.

[9] Nallapati, R., Zhou, B., Santos, C.N.dos, Gulcehre, C., & Xiang,B.(2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. Proceedings of CoNLL.