Abstractive Summarization for Amazon Reviews

Lu Yang
Stanford University
lucilley@stanford.edu

Abstract

This paper focuses on feed-forward neural network with attention-based encoder to solve the challenge of abstractive summarization. We also briefly explored the potential of attentive recurrent neural network and recurrent neural network encoder-decoder. Those models were originally proposed to solve similar tasks, such as news articles summarization and machine translation; we modify and extend them to the problem of product review summarization and evaluate their effectiveness using ROUGE and visual inspection of results. In the end, we find the results to be promising.

1 Introduction

Websites such as Amazon and Yelp allow customers to leave reviews for various products. There are usually hundreds of reviews for a single product; each review could be lengthy and repetitive. Therefore automatic review summarization has a huge potential in that it could help customers to make quick decisions on certain products. In addition, summarization could be applied to not only reviews but also other entities, such as emails and news articles, etc.

Automatic summarization is an active field in Natural Language Processing. Summarization techniques can be classified into two types: extractive summarization and abstractive summarization. Extractive methods attempt to summarize articles by selecting a subset of words that retain the most important points, while abstractive methods select words based on semantic understanding, even those words did not appear in the source documents. Although much work has been done in the area of extractive summarizaiton, limited study is available for abstractive summarization as it requires deeper understanding of the text.

In this paper, we trained a feed-forward NN with attention-based encoder for the purpose of abstractive summarization. It was designed for summarization tasks but was applied over Gigawords, an archive of newswire text data. We also briefly investigated in the RNN encoder-decoder, which was most popular for machine translation, but we want to see whether it can generalize to summarization tasks as well. Additionally, we further explored the attentive recurrent neural network as proposed by Chopra et al [3] and qualitatively evaluated the performance.

2 Related Work

There have been several papers studying the topic of automatic summarization over the years; most of them focus on extractive summarization. The very first work is published by Luhn [6] in the 1950s. He proposed a simple idea that some words in a document are more descriptive of its content than others. He also suggested using word counts to find those words, because words that occur often are likely to be the main topic of the document. Of course we now know that words with higher frequency are not necessarily the main topic and thus his approach cannot work perfectly well, but his work shapes much of later research. Later research refines the idea of using raw frequency of words by using TF*IDF weights [8] and log-likelihood ratio test for topic signatures [5].

In contrast, there has been limited study in abstractive sumarization. But Recent advances in neural networks, specifically machine translation, motivate researchers to rethink the problem. There have been several papers in machine translation, such as recurrent neural network encoder-decoder proposed by Cho et al. [2] and an extension of RNN encoder-decoder with attention mechanism by Bahdanau et al. [1] The models we implemented in this project are inspired by those two papers.

3 Approach

In this section, we described in detail the architecture of the models trained, the datasets over which they were evaluated, and how we departed from the original model.

3.1 Problem Statement

Given a piece of review, we would like to generate a shorter version of the review while preserving the sentiment and points. More formally, Let the input sequence of M words be $x_1, ..., x_M$ coming from a vocabulary of size V. We want to generate a shortened sequence $y_1, ..., y_N$ such that $N < M$ and y retains the essence of x. We assume all $y_1, ...y_N$ coming from the same vocabulary.

3.2 Dataset and Data Preparation

We obtained Amazon reviews from Stanford Network Analysis Project (SNAP). The raw dataset includes 34,686,770 reviews spanning all kinds of products: books, video games, toys, music, etc, where each entry contains product, user information, ratings, a plaintext review and a summary.

Because the dataset is so large, it is unlikely that we will be able to feed all of them into our models. In order to iterate fast, We selected one specific product, book, for testing and debugging our model. We utilized 20,626 reviews for training and 3278 reviews for validation. For test please see multi dataset.

However, one product is unlikely to suffice; we are also interested in understanding how our models perform for various products. For this purpose, we mix and match multiple products into one dataset. We utilized 440,324 reviews for training, 66,893 for validation and 14,836 for test. Multi-product data include the following products: "Books," "Electronics," "Clothing Shoes & Jewelry," "Movies & TV" and "Home & Kitchen." Please note that the test data also include two additional out-of-domain products: "Sports & Outdoors" and "Health & Personal Care."

It is useful to know a few statistics of the dataset beforehand. We use those statistics to tweak bucketing for RNN, as you will see in section 5. By querying data using Google BigQuery, we found that interquartile range of number of words per review is 26-64 words, and interquartile range of number of words per summary is 4-7 words. Additionally, the average number of reviews is more than 100 words, which means the review dataset is skewed.

A lot of preliminary work was devoted to preparing the data. We first exclude titles that contain encode non-words like HTML tags and titles that get accidentally truncated to 128 characters. We then split them into training, validation and test set, and massage them into the format we want. Specifically, we remove question marks, semicolon, colon and punctuation, and convert all the characters to lower cases.

3.3 Model

3.3.1 Feed-forward Neural Network with Attention-based Encoder

Scoring Function: from a set Y that contains all possible sentences of length N, the goal is to find an optimal sequence such that the scoring function s is maximized.

$$\arg\max_{y \in Y} s(x, y) \tag{1}$$

For this specific problem, the scoring function is

$$s = \log p(y|x; \theta) \approx \sum_{i=0}^{N-1} \log p(y_{i+1}|x, y_c; \theta) \tag{2}$$

where y_c is previous words defined as y[i-C+1, ..., i] for a window of size C. For $i < 1$, we pad y_c with the start symbol "<s>". The next section describes how to model the local conditional distribution $p(y_{i+1}|x, y_c; \theta)$.

Modeling Conditional Distribution

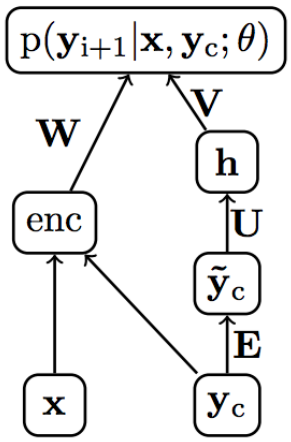

Figure 1: Architecture of feed-forward neural network with attention-based encoder

This feed-forward model, fist proposed by Rush et al. [7], has the following architecture:

$$\begin{aligned} p(y_{i+1}|y_c, x; \theta) &= softmax(Vh + Wenc(x, y_c) + b_2) \\ y_c &= [Ey_{i-C+1}, ..., Ey_i], \\ h &= \tanh(Uy_c + b_1) \end{aligned}$$

where y_c is the context (previous C words). V, W, U are weight metrics and E is an embedding matrix. enc represents an encoder node. We implemented two encoders: bag-of-words encoder and attentive-based encoder.

Model Intuition: We can gain intuitive understanding by comparing the first equation with the following conditional summarization model:

$$\arg\max_y \log p(y|x) = \arg\max_y \log p(y)p(x|y)$$

Roughly speaking, Vh estimates $\log p(y)$ and $Wenc(x, yc)$ estimates $\log p(x|y)$.

Bag-of-Words Encoder: Bag-of-Words Encoder ignores order and relationship between words.

$$\begin{aligned} enc(x, y_c) &= p^T \widetilde{x} \\ p &= [1/M, ..., 1/M] \\ \widetilde{x} &= [Fx_1, ..., Fx_M] \end{aligned}$$

where F is the embedding matrix and p is the uniform distribution. Therefore word with higher frequency gets heavily weighted.

Attention-based Encoder: In contrast to bag-of-words encoder, attention-based contextual encoder exploits the context y_c. The attention mechanism is inspired by Bahdanau et al [1].

$$enc(x, y_c) = p^T \bar{x}$$

$$
\begin{aligned}
p &= softmax(\widetilde{x}Py_c) \\
\widetilde{x} &= [Fx_1, ..., Fx_M] \\
y_c &= [Gy_{i-C+1}, ..., Gy_i] \\
\forall i, \bar{x_i} &= \sum_{q=i-Q}^{i+Q} \widetilde{x}/Q;
\end{aligned}
$$

where F and G are embedding matrices and P is a weight matrix. Q is the smoothing window over review embedding. Intuitively, P is an alignment between the review and context. If the current context aligns well with $x_{i-Q}, ..., x_{i+Q}$, those words are heavily weighted,

Prediction, objective function and gradient descent:We implemented a beam search algorithm for generating predictions. Let the beam size be K. Instead of computing the most likely first word, beam search expands the hypothesis space by computing the K most likely first word. Then to compute the next word, we condition on each of K first word, score all the sequences and again obtain top K of those.

We used stochastic batch gradient descent and cross-entropy loss as objective function.

Word Embedding: Random Initialization vs. GloVe: The original paper initializes the word embedding matrixs randomly. We think initializing with word vector representations that capture semantic and syntactic relationships between words would improve the performance slightly.

Entire review text vs. first sentence of review text: note that one difference between our problem and the original paper is that we use the entire review text, while the original paper only utilizes the first sentence of news articles. We chose this approach because the first sentence of product reviews is usually unable to capture the essence of whole text. Because it would be harder for the model to learn to stop at the end of a sentence, at first we were concerned that the performance may degrade due to this change. But the results proved that this was not an issue.

3.3.2 Recurrent Neural Network Encoder-decoder with Attention Mechanism

Although this product mostly focuses on model 1, we also explored two additional recurrent neural network models. Model 2 and 3 are based on recurrent neural network encoder-decoder proposed by Cho et al. [2]. We will briefly explain them here but won't go in too much details.

Cho et al.[2] first suggested RNN encoder-decoder, which was later extended by Bahdanau et al. [1]. An RNN encoder-decoder consists of two RNNs: the encoder extracts a fixed length representation of the input, and the decoder generates a translation from this representation. Bahdanau et al. [1] extended this model to allow decoder has more direct access to input by implementing an attention mechanism.

Each conditional probability is modeled by

$$p(y_i|y_1, ..., y_{i-1}, x) = g(y_{i-1}, s_i, c_i) \tag{3}$$

Where s_i is a decoder RNN hidden state, and c_i is the context vector computed for each time t. The context c_i is computed as weighted sum of a sequence of encoder hidden states $h_1, ..., h_T$.

$$c_i = \sum_{j=1}^{T} a_{ij} h_j \tag{4}$$

where a_{ij} is softmax of e_{ij}, an alignment model which represents how well input around position j aligns with output at position i.

$$e_{ij} = a(s_{i-1}, h_j) \tag{5}$$

We differed from the original paper in that we used LSTM cells and one-directional RNN. We used argmax for predictions.

3.3.3 Attentive Recurrent Neural Network

The model, recently proposed by Chopra et al. [3], is a simplified version of Model 2. The decoder is an RNN with LSTM cells. Hidden states of RNN are computed as:

$$h_t = g_\theta(y_{t-1}, h_{t-1}, c_t) \tag{6}$$

where y_{t-1} is the previous input word, h_{t-1} is the previous hidden state, and c_t is the output of the encoder. c_t is a weighted average of $x_1, ..., x_M$.

$$c_t = \sum_{j=1}^{M} \alpha_{j,t-1} x_j \tag{7}$$

and α is computed by taking softmax of z_j and h_{t-1}.

$$\alpha_{j,t-1} = \exp(z_j \cdot h_{t-1}) / \sum_{i=1}^{M} \exp(z_i \cdot h_{t-1}) \tag{8}$$

z_j is computed by convolution over embedding of consecutive words. The intuition is z_j is a representation that encodes the position of word and the context in which it appears in the sentence.

Similar to Model 1, we again relied on beam search to give us the most plausible summary.

4 Experiment

4.1 Model 1: Feed-forward Neural Network with Attention-based Encoder

We implemented this model from scratch using Tensorflow. The hyperparameters we use for both bag-of-words encoder and attention-base encoders are: batch size = 64, article embedding size (P) = 200, embed size (D) = 64, hidden size (H) = 128, window size (C) = 5 and smoothing window length = 5. For prediction we use beam size 100.

We used Adam Optimizer, which computes adaptive learning rates for each parameter. We set the learning rate to be 0.001, epsilon 0.0005. Adam optimizer outperformed vanilla gradient descent optimizer in that it converges faster.

We trained the model on AWS EC2 g2.2x large, which took roughly 45 minutes.

Quantitative Analysis: For evaluation, we use ROUGE-1, ROUGE-2 and ROUGE-L [4]. ROUGE is a package of standard metrics for summarization tasks; it measures similarity between reference summary and test summary. **ROUGE-N** is recall-oriented measure; it divides the number of matching n-grams by the number of total n-grams in the reference summary.

$$ROUGE - N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)}$$

ROUGE-L compares the longest common sequence (LCS) between reference summary and test summary. It eliminates the need to pre-define n-gram length. The intuition is that the longer the LCS is, the more similar those two text are. Given Xreference summary X of length m and test summary Y of length n, it computes an F-measure:

$$R_{lcs} = \frac{LCS(X,Y)}{m}$$
$$P_{lcs} = \frac{LCS(X,Y)}{n}$$
$$F_{lcs} = \frac{1 + \beta^2 R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

Below we present results. For discussion please see section 5.

Table 1: Perplexity for Bag-of-words Encoder

	Training	Validation	Test
Book	31.80	77.49	97.05
Multi	27.53	52.87	52.66

Table 2: Perplexity for Attention-based Encoder

	Training	Validation	Test
Book	41.62	86.63	106.06
Multi	23.32	51.68	51.93

We used attention-based encoder for ROUGE, as it yields lower perplexity for various products.

Table 3: ROUGE using Attention-based Encoder

	ROUGE-1	ROUGE-2	ROUGE-L
Book	82.56	77.86	82.30
Health and Personal Care	83.42	78.79	83.29
home and Kitchen	84.01	79.20	83.72
Sports and Outdoors	82.86	78.26	75.03
Movies and TV	82.58	78.15	74.95
Clothing, shoes and jewelry	83.46	78.60	83.13

* Please note that we think ROUGE measurement is off, as by visual inspection we don't think the results are as high as 80%. We used ROUGE software to compute the metrics; there might be some caveats that we missed.

Qualitative Analysis: we visually inspected the results and found a few interesting examples:

> **Original Summary (O)**: "tents architecture of the nomads"
> **Review (R)**: "an excellent source of historic nomadic tent information this book is considered to be one of the leading sources for available information about nomadic tents and their construction full of diagrams line illustrations and construction details it covers a wide geographic area examples of lifestyles and customs help one understand how the cultures needs drove the implementation of practical designs an excellent read for anyone serious about tent history construction or nomadic lifestyles"
> **Summary (S)**: "an excellent book for anyone serious about living and culture"

The model understands this is a book and catches the keyword "cultures" and "lifestyles" from the summary. But It failed to understand this book is about "nomads."

Another example trained on multi-product data yields pretty good results:

> **Original Summary (O)**: "too tight on waist and uncomfortable"
> **Review (R)**: "the elastic on the waist is tooo small and tight i have a smaller waist and was surprised as i typically dont have these issues"
> **Summary (S)**: "dont waste your money on this elastic is too small"

The model is able to catch negative sentiments; Interestingly, for a lot of reviews with negative sentiments, the output summary starts with "don't waste your money on this."

Most of summaries are on point and quite detailed. But there are some reviews that the model gets confused about. The following is one of them.

> **Original Summary (O)**: "kindle edition is badly formatted"
> **Review (R)**: "this might well be a great read however the kindle edition is so badly formatted as to make it almost unreadable it looks as if the text has been scanned in as pictures and the resulting font size is so small as to necessitate using a magnifying glass definitely not suitable for older readers who need reading glasses"
> **Summary (S)**: "this is one of the best glasses ive ever read"

The model is wrong about both sentiments and keywords.

4.2 Model 2: Recurrent Neural Network Encoder-decoder with Attention Mechanism

Since we spent most of time on implementing Model 1, which turns out to be non-trivial, we only briefly quantitatively analyzed Model 2 and 3 due to time constraints. We based model 2 on Tensorflow seq2Seq model.

Table 4: Perplexity for RNN Encoder-decoder

	Training	Validation
Book	28.15	53.78

Here is a piece of positive example:

> **Original Summary (O)**: "good book for a horrible subject"
> **Review (R)**: "the book seems well written and is informative it is not a great subject but was required for my major very boring"
> **Summary (S)**: "good information for the subject"

However, this model tends to be repetitive and has trouble performing sentence breaking, e.g. "best book on the world of world", or "this is the best book i've ever read this book."

4.3 Model 3: Attentive Recurrent Neural Network

We used tensorflow seq2seq framework but changed a lot of the codes to adapt to this model.

Table 5: Perplexity for Attentive Recurrent Neural Network

	Training	Validation
Book	31.47	33.64

> **Original Summary (O)**: "a must read for church personal growth"
> **Review (R)**: "this is a great book with lots of great information based on bible principals i have highlights all over the book and even shared parts of it with our pastor i highly recommend it if your church is not growing or you feel stagnant in your spiritual growth you need to purchase this book its amazing how simple things can make significant changes in our lives dont miss it"
> **Summary (S)**: "one of the best books ive ever read"

5 Discussion/Conclusion

By comparing table 1 and table 2, we observed that attention-based encoder has higher perplexity for the book dataset than for bag-of-words encoder. This may be due to (1) some intrinsic properties of book reviews make it harder to apply attention on, or (2) human-generated book review summaries are not well-curated enough so that they confused the model. For multi dataset, attention-based encoder outperforms bag-of-words encoder, which is expected.

We also confirmed that this feed-forward neural network can generalize well to product review summarization tasks. Interestingly, we observe that the model is quite conservative, focusing on high-level description rather than specific details ("this is a good book" vs. "the storyline is moving"). In other words, the model tends to generalize over the entire datasets.

We have also seen that although perplexity for Model 2 and 3 are lower than Model 1, results are not necessarily better. Model 2 has trouble performing sentence breaking. Model 3 tends to output very general summary, which could be due to flaws of our implementation of RNN beam search. We reserve improvements on Model 2 and 3 for future work.

6 Future Work

Since this paper mostly focuses on the feed-forward neural network, we haven't had the chance to explore Model 2 and 3 in depth. Immediate next steps would be to run Model 2 and 3 on multi-

product data and evaluate using ROUGE, as book data may be hard to apply the attention mechanism on.

Another intuitive extension of Model 2 is to use bi-directional RNN. Bi-directional RNN is based on the insight that the output at time t may not only depend on previous words, but also future words. Bi-directional RNN allows the model to better exploit the context.

One more extension for Model 2 and 3 is to use deep(multi-layer) RNN which will enhance learning capacities. Since we have huge amount of data, we shouldn't need to worry about the problem of overfitting.

References

[1] Bahdanau, D. et al. (2015) Neural Machine Translation by Jointly Learning to Align and Translate. *Proceedings of the 2015 Conference on International Conference on Learning Representation (ICLR)*.

[2] Cho, K. et al. (2014) Learning Phrase Representations using RNN Encoder Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[3] Chopra, S. & Mozer, M.C. (2016) Abstractive Sentence Summarization with Attentive Neural Networks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

[4] Lin, C. (2004) ROUGE: A Package for Automatic Evaluation of Summaries.*Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004*. Barcelona, Spain.

[5] Lin, C. & Hovy, E. (2000) The automated acquisition of topic signatures for text summarization. *Proceedings of the International Conference on Computational Linguistic*.

[6] Luhn, H. P. (1958) The automatic creation of literature abstracts, *IBM Journal of Research and Development*, vol. 2, no. 2.

[7] Rush, A.M., Chopra, S. & Weston, J. (2015) A Neural Attention Model for Sentence Summarization. *The 2015 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

[8] Salton G. & Buckley C. (1988) Term-weighting approaches in automatic text retrieval. Information Processing and Management, vol. 24, pp. 513 523.

Supplementary Material: Source Code

1. "data/" contains dummy data used for debugging, and scripts to get data and process data.

2. "feed-forward/" contains implementation of Model 1: feed forward neural network.

3. "rnn/rnn_encoder_decoder" contains implementation of Model 2: RNN encoder-decoder with attention mechanism.

4. "rnn/attentive_rnn" contains implementation of Model 3: recurrent attentive neural network.

5. "eval" contains scripts for evaluation using ROUGE.