Opinion Generation using Abstractive Text Summarization Techniques

Sumedh Sankhe, Rajath Kashyap and Soumitra Mishra

Abstract

Text summarization problem has gained a lot of traction in the recent past since it can solve a wide range of problems. Natural language processing has seen a rise of deep-learning based models that map an input sequence into another output sequence, called sequence-to-sequence models which have been successfully applied to many problems. In this project we try to use sequence-to-sequence bidirectional RNNs to tackle the problem of abstractive text summarization. We also explore an unsupervised way to extract summaries initially which we further use in training our neural model.

1 Problem Description

While making a decision to eat-out, going through thousands of reviews for each restaurant is a tedious task. The objective of this project is to simplify and speed-up this task by developing a generic opinion of a restaurant using customer review data from Yelp dataset. This is achieved by reading and understanding the reviews, extracting important features and information from them and condensing this information into small summary portraying major ideas of all the reviews.

Firstly, we designed an unsupervised model that takes all the reviews as input and generates list of phrasal summaries out of which the best one are selected and combined to develop the final summary.
For example, if we are given n reviews for a restaurant called ”The Friendly Greek Restaurant ” as:
1. From the outside the place looks like it might be good. The atmosphere inside is nice as well but the food to me tasted no different than packaged Greek food you can get at the grocery store and cook at home in a fly pan.
2. The fountain pop was watered down too what a horrible way to try and cut costs. What can you say about a fast-food Greek restaurant.
...
...
n. It definitely isn't as authentic as some of the pricier Greek restaurants in Toronto, but it is tasty.
Our unsupervised model will generate a summary as follows: -
”greek food really isnt authentic greek food pricier greek food”

Since the summaries generated were not fluent enough, we build an LSTM model which generated as summary as follows for the above example: -
”Bad food quality Greek Greek tastes like processed food”

2 Previous Work

- In 2008 S.R.K. Branavan et. al [1] demonstrated a new method for leveraging free-text annotations to infer semantic properties of documents. Due to the dramatic growth of semi-structured user generated online content. An example of such content is product reviews, which are often annotated by their authors with pros/cons keyphrases such as “a real bargain” or “good value.” To exploit such noisy annotations, they simultaneously found a hidden paraphrase structure of the keyphrases, a model of the document texts, and the underlying semantic properties that link the two. This allowed them to predict properties of unannotated documents. They implemented a hierarchical Bayesian model with joint inference, which increases the robustness of the keyphrase clustering and encourages the document model to correlate with semantically meaningful properties.

- In 2012 Ganesan et.al [2] proposed an Unsupervised with a heuristic algorithm to generate ultra concise summaries of opinions, this method treats the problem as an optimization problem where and measures the representation base on a modified mutual information function and an n-gram based

language model

- In 2016 Nallapati et. al modelled abstractive text summarization using Attentional Encoder-Decoder Recurrent Neural Networks, they proposed several models that address critical problems in summarization that are not adequately modeled by the basic architecture, such as modeling key-words, capturing the hierarchy of sentence-to-word structure, and emitting words that are rare or unseen at training time.

3 Methodology

3.1 Pre-processing

The data consists of 5 json files of which only the following two were utilized for this task:

- **yelp_academic_dataset_review.json**: contains reviews for every business-id
- **yelp_academic_dataset_business.json**: contains the mapping of business to business-id

The business json file has information of 192,609 businesses, these businesses fall into a variety of categories ranging from Salons, Spas, Home Services, Restaurants and other leisure activities. For the purpose of this experiment we decided to keep only establishments that are classified as restaurants. This resulted in reducing our total tally of restaurants to 59,371. This blob of review's and business id's was further broken down to individual documents containing only reviews for a particular type of restaurant. Pre-processing on these individual files was further carried out to standardize the texts which consisted of using lower case, using the tweet vectorizer functionality of nltk to remove redundant words. The sentences were tokenized to add an end of sentence tag to help recognize sentences.

For the Bi-directional LSTM implementation the pre-processing of data changes, since we are interested in the sentence and grammatical structure of the text we do not remove stopwords from the text. The text is transformed to lower case, number are preserved by transforming them to symbols. The summaries are similary cleaned and one hot encoded.

3.2 Features

The first approach we use to generate summaries is partly unsupervised with language model. Here we try to extract summary using the bag of words approach. So we use highest frequency words/unigrams (adjustable parameter) from the reviews which forms the first set of features. Using this we build the our base feature the seed bigrams. Seed bigrams are a combination of all the bigrams that can be formed using the top frequency unigrams.

For the sequence-to-sequence model using the bag of words approach does not work since we need to generate a summary which needs to keeps track of the context of the reviews and also it is important to maintain the ordering of words. To try to improve our text summarization we tried using pre-trained word embeddings of the google news vectors, this resulted in summaries having meaningless sentences that did not align with the reviews that were being put in but more with the news articles the embedding were trained on. Hence we trained our own word embeddings using the word2vec model, entire corpus excluding the test set were included in the training of the word embeddings.

3.3 The unsupervised approach

The outcome of this model is to get concise and informative summary described in Ganesan et. al [2] that can give a generic opinion about the restaurants that is mentioned in the reviews. To get the summary out we use three important scores to keep a check on the model i.e. Readability, representativeness and the similarity. Readability (S_{read}) is how well formed and grammatically correct the summaries are. This is very important for making sense of the summary. For calculating this we have used the Flesch reading score. Representativeness (S_{rep}) is selecting text that covers the major opinions. This is calculated using the Pointwise mutual information (PMI) between the words. This is calculated as:

$$S_{rep}(w_1 ... w_n) = \frac{1}{n} \sum_{i=1}^{n} pmi_{local}(w_i)$$

where pmi_{local} is the local PMI score which is given by

$$pmi_{local}(w_i) = \frac{1}{2n} \sum_{j=i-C}^{i+C} pmi^{'}(w_i, w_j), i \neq j$$

Here C is the context window size, which means the C neighbouring words on the left and right of the a word w_i. The PMI score gives information about how the words are associated between each other. For the phrase "service staff friendly", the pmi is for the word service is calculated with staff. For the word staff, the score is calculated between staff and service and also

between staff and friendly. Finally score for the word friendly it is calculated with staff. The pmi' between two words is calculated using the formula:

$$pmi'(w_i, w_j) = \log \frac{p(w_i, w_j).c(w_i, w_j)}{p(w_i).p(w_j)}$$

where c represents the count function and p represents the probabilities.

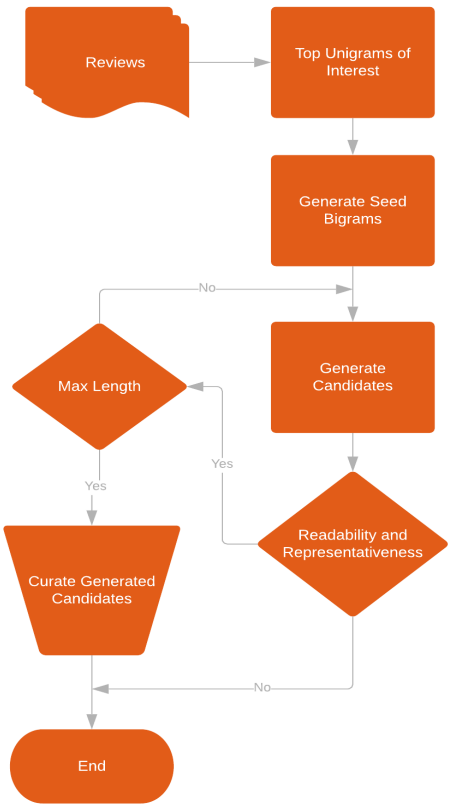

Figure 1. Unsupervised Model Flowchart

The similarity is used between the phrases in the potential candidates and among the candidates in the list to remove the redundant summary candidates. We use the cosine similarity between the phrases to determine the redundant candidates (different to jaccard similarity used in Ganesan et. al [2]). The cosine similarity gives a better similarity score as it is a vector and has a sense of direction to it unlike jaccard similarity.

To generate the summary we use a recursive approach to keep building the summary. In each recursion we try to eliminate the candidates by checking if they are above a treshold score which is the sum $S_{read} + S_{rep}$.

The seed bigrams which is the starting point of our summary generation is formed by taking the most frequent words in each summary. To grow these bigrams to seed n-grams which finally gives us the potential summary we concatenate the seed bigrams keeping in consideration the above mentioned scores at each recursion. We also have an limit on the number of words must be in the summary, above which our recursion stops.

This method gave us average results, but we could not use it in our training process as stated before. This method would have worked well if we had a better quality text or reviews which are written by an expert. Since our reviews were written by users and there was noise and polarity in the reviews, this method did not work as expected. The results made sense when read by a human giving an opinion but it was not suitable to use it for our RNN model.

3.4 Bi-Directional Sequence to Sequence RNN

Our approach is to implement a simple RNN encoder-decoder framework as describe by Cho et. al [5] In the Encoder–Decoder framework, an encoder reads the input sentence, a sequence of vectors $x = (x_1, ..., x_T x)$, into a vector c. The most common approach is to use an RNN such that

$$h_t = f(x_t, h_{t-1})$$

and

$$c = q(h_1, ..., h_T x)$$

, where $h_t \in R$ n is a hidden state at time t, and c is a vector generated from the sequence of the hidden states. f and q are some nonlinear functions. Sutskever et al.[6] used an LSTM as f and $q(h_1, ..., h_T) = h_T$, for instance. The decoder is often trained to predict the next word $y_t^{'}$ given the context vector c and all the previously predicted words $y_1, ..., y_{t-1}$. In other words, the decoder defines a probability over the translation y by decomposing the joint probability into the ordered conditionals:

$$p(y) = \prod_{t=1}^{T} p(y_t | y_1, ..., y_{t-1}, c)$$

where $y = (y_1, ..., y_{Ty})$. With an RNN, each conditional probability is modeled as

$$p(y_t | y_1, ..., y_{t1}, c) = g(y_{t-1}, s_t, c)$$

where g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t, and s_t is the hidden state of the RNN.The usual RNN, described above reads an input sequence x in order starting from the first word x_1 to the last one x_{Tx}.

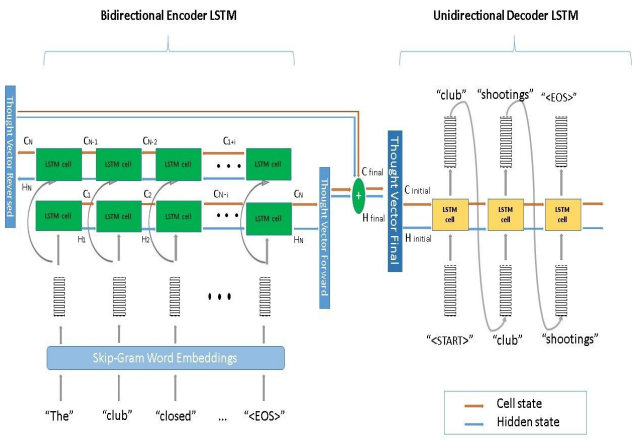

Figure 2. Model Architecture

However for our problem we need annotations for words that preceding our y_t as well as following our word, this task is successfully accomplished by a BiRNN which consists of forward and backward RNN's. The forward RNN f reads the input sequence as it is ordered (from x_1 to x_{Tx}) and calculates a sequence of forward hidden states $(h_1, ..., h_{Tx})$. The backward RNN $f^{'}$ reads the sequence in the reverse order (from x_{Tx} to x_1), resulting in a sequence of backward hidden states $({h_1}', ..., {h_{Tx}}')$. We obtain an annotation for each word x_j by concatenating the forward hidden state h_j and the backward one $h_j^{'}$, i.e., $h_j = h_j^T : h_j^{T\prime}$. In this way, the annotation hj contains the summaries of both the preceding words and the following words. Due to the tendency of RNNs to better represent recent inputs, the annotation h_j will be focused on the words around x_j . This sequence of annotations is used by the decoder and the alignment model later to compute the context vector

4 Experimental Setup

4.1 Datasets

From the Yelp data-set, we considered reviews for 200 restaurants having an average of 8-10 reviews each. The data-set was divided as follows: -
1. 70% for training.
2. 20% for validation.
3. 10% for testing.
To train the model, manually written summaries were used as target.

4.2 Evaluation

We used ROUGE scores in order to evaluate our summaries leveraging manually written summaries as reference.

- ROUGE-1 refers to the overlap of 1-gram (each word) between the system and reference summaries.
- ROUGE-2 refers to the overlap of bigrams between the system and reference summaries.

4.3 Baseline

To assess how well our new models work, we made use of an pre-trained off the shelf model that is available in the textgenrnn library. We passed the individual reviews file for a restaurant named Gordon Ramsay Steak to the neural network.
The neural network was a bi-directional RNN with 64 nodes in it and was pre-trained on a large corpus of data. We ran it only for 1 iteration on our file which gave us following result:

```
###################
Temperature: 1.0
###################
Foie:  when it was all I an also ordered amazing... pepperse 13,
enjuy is notimate your routhel dish was agoly. <end>  <start> The
Strain Boup i he saw I have ever wanted it enortry this last I
wasn't anioce was really answardly. This is the best did no the
polic though too wsharing the other flavo
```

The summaries generated by this model were unreadable and did not make any sense. By glancing at the summary that was generated the number of training epoch were clearly not enough the network did create some garbled summary.

4.4 Results

The mean ROUGE scores against manually generated summaries are as follows:

- **Unsupervised Model** -
  - ROUGE 1 – 0.411
  - ROUGE 2 - 0.277
- **LSTM Model** -
  - ROUGE 1 – 0.212
  - ROUGE 2 - 0.059

Manually Generated Summaries	Model 1 Summary	Model 2 Summary
Friendly staff and good food, serves great beer	food pretty good service staff friendly great beer cheap food	Steak burger and beer at price cheap at ### reasonable
Non authentic greek food restaurent you can avoid unless you do not want to eat packaged food.	greek food really isnt authentic greek food pricier greek food	Bad food quality Greek Greek tastes like processed food.
Bagel shop which serves the worst bagel. Avoid visiting the shop.	cream cheese excited favourite type bagels average best go virtually anywhere else	Disappointing Bagel in bagel shop and cream cheese is soggy bagel

Table 1. Example Generated Summaries

5 Future Work

The text summarization task works better if we use an attention model rather than a simple rnn. By adding an attention mechanism we can give weightage and concentrate on the words that are of more importance. As explained in Nallapati et. al [7], the attention model gives a promising result.

References

[1] S. R. K. Branavan, H. Chen, J. Eisenstein, and R. Barzilay. Learning document-level semantic properties from free-text annotations. In In Proceedings of ACL, pages 263–271, 2008,

[2] Kavita Ganesan, ChengXiang Zhai, and Evelyne Viegas. 2012. Micropinion Generation: An Unsupervised Approach to Generating Ultra-concise Summaries of Opinions. In Proceedings of the 21st International Conference on World Wide Web (WWW '12). ACM, New York, NY, USA, 869–878.

[3] Alexander M. Rush, Sumit Chopra, Jason Weston. 2015.A Neural Attention Model for Abstractive SentenceSummarization, Facebook AI Research / Harvard SEAS,September 2015

[4] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Bing Xiang, Çaglar Gulçehre, Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond, Aug 2016

[5] Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014a). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014). to appear.

[6] Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NIPS 2014).

[7] Proceedings of NAACL-HLT 2016, pages 93–98, San Diego, California, June 12-17, 2016. c 2016 Association for Computational Linguistics Abstractive Sentence Summarization with Attentive Recurrent Neural Networks, Sumit Chopra, Michael Auli, Alexander Rush

[8] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graphbased approach to abstractive summarization of highly redundant opinions. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING '11). Association for Computational Linguistics, Stroudsburg, PA, USA, 340–348.

[9] Regina Barzilay and Lillian Lee 2003. Learning to Paraphrase: An Unsupervised approach using multiple sequence alignment. In Proceedings of HLT/NAACL 2003. Department of Computer Science Cornell University Ithaca, NY: 14853-7501.

[10] Josef Steinberger, Karel Jezek 2007. Evaluation Measures for Text Summarization. Department of Computer Science and Engineering University of West Bohemia in Pilsen, 306 14 Plzen, Czech Republic.

[11] https://github.com/DeepsMoseli/Bidirectiona-LSTM-for-text-summarization-

[12] https://www.kaggle.com/yelp-dataset/yelp-dataset,