Aleksander Petuskey
3032750086
17petuskey@berkeley.edu

NLP Project - Yelp Review To Rating
CS 182, Spring 2021

2021-05-13

1 Introduction

Yelp is a popular online directory for discovering local businesses ranging from restaurants to barbershops. Yelp relies on user reviews to inform its users of the quality of businesses in the local area. Users can leave reviews, star ratings, and photos of their experience with each business they visit. This is essential to Yelp's business model because they can provide advice and consultation to businesses on how exactly they can improve their services while also informing their users on the quality of service they can expect to receive at each particular location.

Yelp periodically publicly releases subsets of its reviews as datasets with accompanying challenges for the research community. The specific dataset used in this assignment yelp review training dataset and can be found at this link. In this dataset, each data point consists of a review id, text string containing the written review, and a corresponding star ranging from 1-5 rating for that review.

Yelp Review Example

Review Id	Text	Stars
üqxkO6B6w_sIDSAGr0k_0A"	Üne institution du caf̃0e9"	2.0

The goal of this project is to predict the star ratings of a given review. By using machine learning models to extract features associated with predicting star ratings we can better understand what contributes most to the actual star review given. Success measures of this project will be based on two calculations Average Star Error (Average Absolute offset between predicted and true stars) and Exact Match (Number of exactly predicted star ratings / total samples). In this project, I will explore a variety of deep learning architectures and techniques to try and get the lowest Average Star Error and highest Exact Match. I will start out by creating a baseline Bidirectional LSTM Concat then explore other architectures such as RNN's and pretrained transformer models.

2 Literature Survey

When I first examined this problem, to get a better understanding of sentiment analysis in NLP I read a brief tutorial titled, Än easy tutorial about Sentiment Analysis with Deep Learning and Keras", by Sergio Virahonda on Towards Data Science. While the tutorial itself is quite basic it introduced me to some important subjects which I continued to learn about and research while completing this project. His introduction to word embeddings lead me towards learning about GloVe: Global Vectors for Word Representation. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. I ended up experimenting with this briefly before pivoting as shown later in the report. Additionally, the Towards Data Science article used a Bidirectional LSTM which I implemented on my own as a baseline metric for my report. Following this article, I came across a paper by Tomas Mikolov titled, "Recurrent neural network based language model."where he uses a recurrent neural network (RNN) based language models to improve the BUT English meeting recognizer. This lead me to experiment with RNN's and implemented a bidirectional RNN and an attention-based RNN. After implementing these the results weren't as good as I want so I experimented with pretrained transformer models. The article "BERT, RoBERTa, DistilBERT, XLNet — which one to use?"by Suleiman Khan, Ph.D. gave me an introduction to a few popular transformers. This lead me to explore more about XLNet where I read papers such as "XLNet: Generalized Autoregressive Pretraining for Language Understanding"to learn more about the architecture.

3 Background

To understand my work one would first have to understand my baseline model a Bidirectional LSTMs which are an extension of traditional LSTMs that can improve model performance on sequence classification problems. The idea of Bidirectional Recurrent Neural Networks (RNNs) is pretty simple, it involves duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second.

Once my baseline model is understood the next relevant topic in my report is RNNs. One should be familiar with the two well-known issues of RNN's the exploding and the vanishing gradient problems. This

was one of the issues as to why I pivoted to my last type of model transformers. I primarily used XLNet for this part of the project. XLnet is an extension of the Transformer-XL model pre-trained using an autoregressive method to learn bidirectional contexts by maximizing the expected likelihood over all permutations of the input sequence factorization order.

4 Methods & Approach

4.1 Data Processing

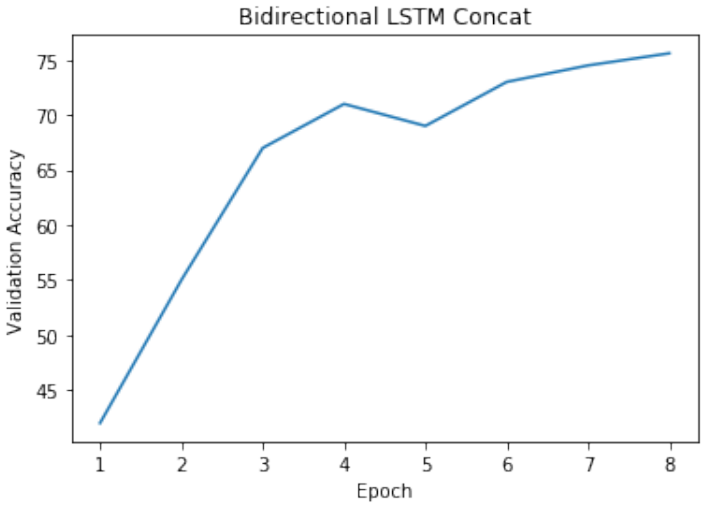
Before approaching the problem, I noticed that the Yelp views written had a wide range of symbols, slang, and sentence structure. Examining the amount of unique words totaled over 500,000. To cut down on this amount I decided employ a few different techniques. The most basic on to start was to get rid of miscellaneous symbols that held little value to the meaning of the sentence. Following this I used stemming to help standardise words to their base root or stem. The idea behind this was to help with classification and information retrieval by reducing inflectional forms and derivationally related forms of a word to a common base form.

Stemming Example: cat, cats, cat's, cats' $\Rightarrow$ cat

Following this, my initial approach was to further cut down on this vocabulary. I started further cut down the unique word total by using word embeddings to combine closely associated words. I used GloVe Word embeddings to accomplish this. As a result, of this procedure, I cut down the number of unique tokens/words to under 150,000. However, after testing the results of this I decided it didn't make much sense because my approach primarily used transformers pretrained with specific embeddings in mind. So I decided not to implement embeddings in my final approach, just sticking with the original preprocessing.

4.2 Baseline Model

I first started by trying a get a working benchmark. To do this I used a Bidirectional LSTM Concat network. In my network the input is a matrix of bitch batch size x 2000. 2000 is what I set my max sequence length to be and used padding when shorter. The output of this network is a the final forward and backward states concatenated with one another.



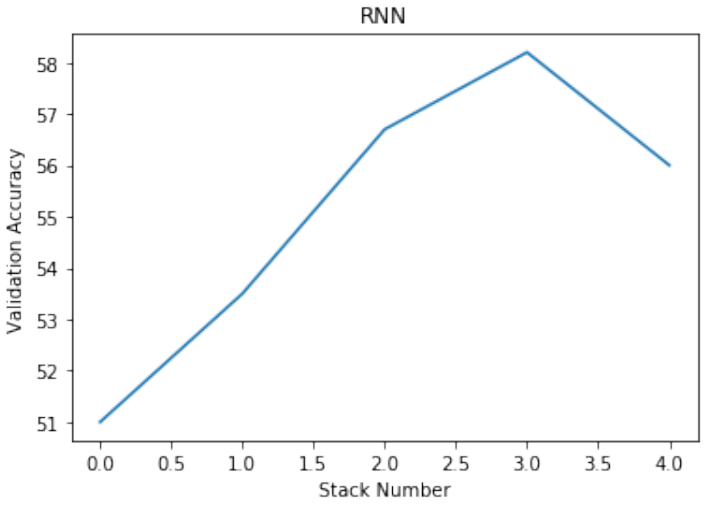
4.3 Recurrent Neural Network (RNN)

I decided to expand on my baseline model by building a couple of variations of RNN architectures. I started out by using a bidirectional RNN. As seen from the background section of the paper a bidirectional RNN consists of a forward and a backward RNN structure. In the forward RNN, the input sequence is arranged from the first input to the last, the model computes a sequence of forward hidden states. The backward RNN takes the input sequence in reverse order, resulting in a sequence of backward hidden states. Finally to get a result I then concatenate output from both RNNs to make the final prediction.

My initial results for my bidirectional RNN were not very accurate. In my bidirectional architecture, all of the information must be stored in the last layer so when a long input is passed in it must capture all the information from the previous states. To solve this it is common to have an attention-based RNN. Attention is a

mechanism combined in the RNN allowing it to focus on certain parts of the input sequence when predicting a certain part of the output sequence, enabling easier learning and of higher quality. In my case, each forward and backward layer is concatenated in a single output vector and transformed into a scalar based on the given attention weights. The scalar from each state is concatenated into a new vector for a final prediction. As a result, I got slightly better results compared to my initial bidirectional RNN. However, after some trial and error, I still wasn't achieving as good of Average Star Error and Exact Match scores as my baseline Bidirectional LSTM . I decided to then pivot and try some pretrained transformer models instead.

As seen in the chart below the RNN did not preform as well as my baseline. In the chart below a 0 stack number was just a bidirectional RNN without attention.



4.4 Transformers Model

Following this baseline approach and unimpressive RNN trails, I wanted to experiment with some applied transformer based models. I first chose BERT language representation model, where deep bidirectional representation from unlabeled text is pretrained on both left and right context for masked language model and next sentence prediction tasks. I used the BERT version with 300M+ parameters. I used a maximum sequence length of 2000 padding similarly as the Bidirectional LSTM, 5 epochs, and a batch size 32. I then expanded on my pretrained tranformer models by experimenting XLNet a generalized auto-regressive pre-training method improved from BERT. I was able to get slightly better results as seen in the results section below.

5 Results

Architecture	Yelp 5 MAE	Yelp 5 Exact Match	Yelp 8 MAE	Yelp 8 Exact Match
XLNet	0.654	0.404	0.448	0.658
RNN	0.78	0.23	0.561	0.451
Bidirectional LSTM	0.71	0.34	0.497	0.542

6 Conclusion

In this project I started out with getting a Bidirectional LSTM Concat network. I then began experiment with RNNs, however, I didnt achieve as great as sucess I expected. I then pivoted to some pretrained transformer model including Bert and XLNet. I ended up using XLNet as my final submission as it achieve the best results on the two challenge set released.

7 Team Contributions

I worked on this project solo so everything was done by Alek Petuskey.

8 References

1. Mikolov, Tomas, et al. "Recurrent neural network based language model.Ïnterspeech. Vol. 2. 2010.

2. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
3. https://medium.datadriveninvestor.com/attention-in-rnns-321fbcd64f05
4. https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8