Evaluation of Convolution LSTM for Sentiment Analysis

by Thorny Seahorses: Morgane Pizigo (mpizigo), Grace Chen (gchen76), Daniel Zhu (dzhu36)

Introduction

Consumer reviews on social media are a goldmine of user data, and heavily influence business and brand image. However, it can be difficult to filter and extract some form of meaningful, large-scale feedback from them without the help of deep learning models. Sentiment analysis (a classification task) helps solve this issue by classifying reviews into predefined categories: subjective/objective, positive/neutral/negative… This approach has previously been used to filter through movie reviews and even predict political leanings of Twitter users with much success; however current models are struggling to keep up with the exponential increase in the amount of data available to handle .

The paper we have chosen outperforms previously implemented models for sentiment analysis by blending two deep learning architectures and drawing upon the benefits of each. This hybrid approach dubbed Co-LSTM combines the precise local feature selection of a CNN with the effective sequential analysis of an LSTM. This allows for better scalability with big social data and makes for a more adaptable, non domain-specific model. This new flexible approach helps us take full advantage of current social big data when performing sentiment analysis.

Methodology

Data

We used three of the four datasets cited in the paper (as the fourth one was no longer available). These include the "Large Movie Review Dataset" known as the IMDB dataset, which contains 50,000 highly polar movie reviews; the Airline Sentiment Review dataset containing 55,000 positive, neutral, or

negative Twitter reviews from 6 major US airlines scraped from Twitter from January 2013 to February 2014 by the Wan and Gao (2015) paper; and the US Presidential Election Dataset with 13,871 positive, neutral, or negative Twitter posts scraped from Twitter in 2016. We also used a fourth "experimental" dataset to test the effectiveness of our model, namely an Anime Review Dataset containing 130,000 reviews with an associated numerical rating, scraped from myAnimelist.com.

Preprocessing consisted of denoising the csv datasets, relabeling labels to 0 and 1 for negative/positive sentiment, and formatting the text samples. Two datasets contained rows with "neutral" labels, which were excluded, and one dataset contained a numerical rating that was converted to "positive" and "negative" sentiment (>7 for positive, <3 for negative).

Model

Our model primarily consists of a word embedding layer, a convolution layer, and an LSTM layer, in that order, as those layers are the three central components of the model in our inspiration co-LSTM paper.

The word embeddings creates embedding representations of the various words in one of four input datasets – a dataset of movie reviews, a dataset of election-related tweets, a dataset of airline reviews, and an experimental dataset not mentioned in our inspiration paper that contains anime reviews. The convolution layer identifies critical features in the datasets' sentences for the model's eventual determination of whether a text sentiment is positive or negative. The LSTM layer learns additional relationships between words in input sentences that help solidify the model's final prediction. The output of the LSTM layer is inputted into a fully connected layer with a sigmoid activation to determine the probabilities of a movie review being positive or negative. However, our model also contains a batch normalization layer and a dropout layer to help combat overfitting. We additionally use an Adam optimizer when training our model.

Specific hyperparameters of those layers, such as the number of filters applied to the CNN layer as well as whether to use same or valid padding, have the same values as the hyperparameters used in the inspiration paper, namely:

batch size	64	**hidden size**	256
activation	sigmoid	**learning rate**	0.001 (Adam)
epochs	5	**dropout rate**	0.5

We trained our model by splitting our pre-processed data into 70% training data and 30% testing data. Our model's performance was updated based on gradient backpropagation from our final sigmoid layer up until our convolution layer, where the parameters of all subsequent layers were modified through gradient descent. We used binary cross-entropy loss as our loss function – our model will evaluate inputs as being positive reviews or negative reviews. We also printed out the accuracy of our model using Tensorflow's binary accuracy function.

Files

Our program contains 3 files:

- a preprocessing.py file : it preprocesses all files (denoising, replacing labels by 0 or 1 for negative or positive sentiment, truncating and lemmatizing as needed), then computes the word embeddings for each dataset and saves them in a pickled file.
- a model.py file : contains our co-LSTM's architecture and the train and test functions. The file repeatedly adjusts the model's parameters by passing in data through the model's various layers.
- an assignment.py file : it takes in a —data_source argument (« movies », « elections », « airlines », « anime ») depending on the dataset to run. It feeds the data into our co-LSTM model, trains the model using specific hyperparameters, and then prints out the training and testing accuracies.

At first, we suspected that the preprocessing stage would be the hardest part about implementing the model due to the vast amount of data we needed to modify, but throughout the course of implementing the model, we discovered that fine-tuning the model to prevent overfitting and ensure correct shapes was more difficult.

Results

Running our model on the movie reviews dataset consistently produces training accuracies of around 0.95 and testing accuracies of around 0.81. Using binary-cross entropy, our final training loss was around 0.16 while our testing loss was around 0.54. Similarly, the anime reviews dataset consistently produces training accuracies of around 0.90 and testing accuracies of around 0.79, as well as training

losses of 0.26 and 0.50. The election tweets dataset consistently produces training accuracies of around 0.90 and testing accuracies of around 0.86, as well as training losses of 0.24 and testing losses of 0.35. The airline reviews dataset consistently produces training accuracies of around 0.95 and testing accuracies of around 0.92, along with training losses of 0.13 and testing losses of 0.23.

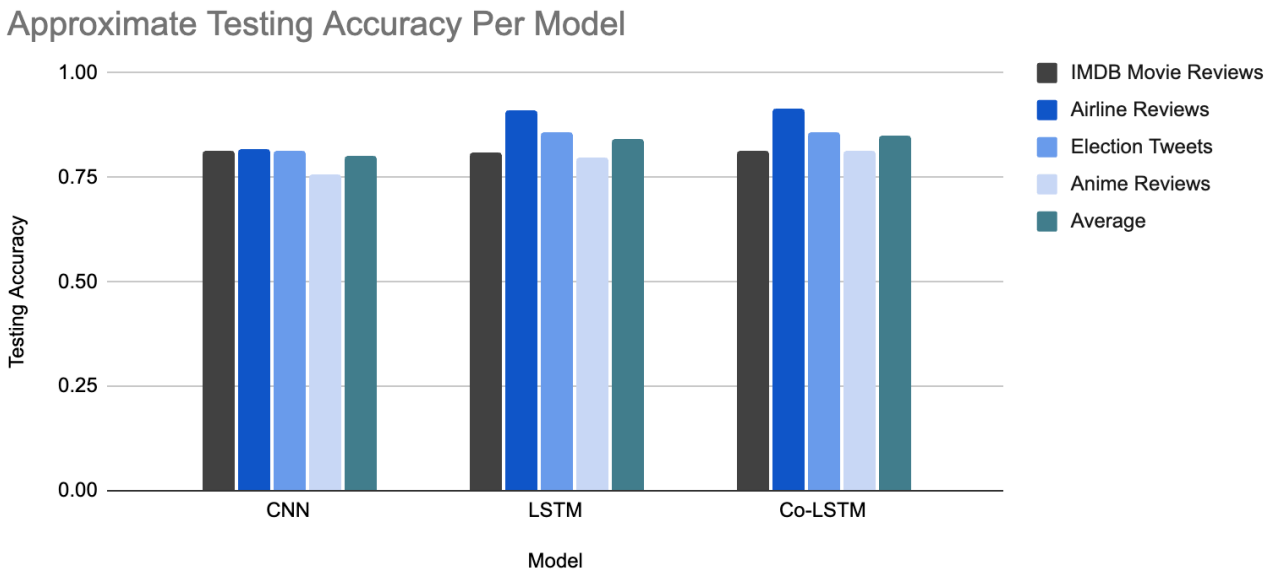
Loss table on co-LSTM model

Epoch	IMDB reviews	Airline reviews	Election tweets	Anime reviews
0	0.451	0.346	0.451	0.449
1	0.314	0.197	0.323	0.357
2	0.249	0.161	0.290	0.291
3	0.196	0.144	0.265	0.236
4	0.160	0.126	0.243	0.196
Test	0.542	0.225	0.346	0.304

Accuracy table on co-LSTM model

Epoch	IMDB reviews	Airline reviews	Election tweets	Anime reviews
0	0.779	0.852	0.806	0.788
1	0.870	0.921	0.869	0.845
2	0.904	0.938	0.883	0.880
3	0.928	0.946	0.895	0.910
4	0.947	0.952	0.903	0.929
Test	0.812	0.915	0.860	0.815

We also compared the performance of the CoLSTM model with LSTM and CNN models. The LSTM model had an average testing accuracy of 0.844 across all four datasets. The CNN model had an average testing accuracy of 0.801 across all four datasets. The Co-LSTM had a slightly higher average testing accuracy of 0.851 across all four datasets, indicating that the Co-LSTM model performed slightly better than the LSTM and CNN models.



Challenges

One of our biggest challenges was modifying the shapes of our model layers' outputs. Since we batched our inputs, we had to ensure we accounted for batching when reshaping our outputs so that all inputs to subsequent layers would not result in a shape error. Moreover, we noticed that the shape of the final output of our model's call function (which was 3-dimensional) did not match the shape of our labels (1-dimensional), which didn't crash the model but instead produced stagnant losses around 0.51 (and similarly stagnant accuracies) throughout both training and testing. Therefore, we had to reshape the call output for our model to effectively train. Another significant challenge we encountered was increasing the accuracy of our model. Not only did we struggle to figure out how to fix our stagnant loss issue, we struggled to reduce overfitting for anime reviews and movie reviews datasets. We ended up creating a batch normalization layer and a dropout layer to help combat overfitting, but the testing accuracies for those two datasets remain considerably lower than the training accuracies. We also experimented with various hyperparameter values to see which values would increase our model's testing accuracy, eventually settling on a batch size of 64 and a hidden size of 256, but since our model took considerable time to train on our datasets (notably, the anime reviews dataset took more than 10 minutes to be trained on), this process was considerably time-consuming.

Reflection

Ultimately, this project was challenging but rewarding. We learned that training a model to perform a task on many different datasets can be a difficult problem, and preprocessing those datasets can be hard as well! Although our proposed model architecture didn't perform as well as we thought it would and had greatly varying results between each dataset, we were still able to demonstrate the viability of our model's hybrid architecture.

We were able to reach all of our base and target goals, including an 80% minimum accuracy across all of our datasets. We were able to reach our stretch goal of using a new dataset (anime reviews) not mentioned in the original paper, but we weren't able to implement a CoTransformer architecture successfully. As such, additional improvements had we had more time include the implementation of the CoTransformer architecture alongside other alternate models, reducing overfitting for greater accuracy, and possibly testing the model with more sentiment categories (sorting into neutral sentiment as well).

Some pivots we made along the way were shifting from 1D to 2D convolution (while 1D is typically used for text, we followed the paper in its 2D convolution with padding approach), as well as slightly changing the model architecture by adding dropout and batch normalization layers to combat the overfitting we were facing. Overall, this taught us flexibility and creativity, even within the limited scope of reproducing a paper's model. It also taught us the importance of details such as hyperparameters, as the paper's lack of clarity led to us having very different results!

References

"Airline Twitter Sentiment - Dataset by Crowdflower." *Data.world*, data.world/crowdflower/airline-twitter-sentiment.

"Anime Dataset with Reviews - MyAnimeList." *Www.kaggle.com*, www.kaggle.com/datasets/marlesson/myanimelist-dataset-animes-profiles-reviews.

Behera, Ranjan Kumar, et al. "Co-LSTM: Convolutional LSTM Model for Sentiment Analysis in Social Big Data." *Information Processing & Management*, vol. 58, no. 1, Jan. 2021, p. 102435, https://doi.org/10.1016/j.ipm.2020.102435. Accessed 29 Nov. 2020.

"First GOP Debate Twitter Sentiment." *Www.kaggle.com*, www.kaggle.com/datasets/crowdflower/first-gop-debate-twitter-sentiment?resource=download. Accessed 11 May 2024.

"IMDB Dataset of 50K Movie Reviews." *Www.kaggle.com*, www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.

Wan, Yun, and Qigang Gao. "An Ensemble Sentiment Classification System of Twitter Data for Airline Services Analysis." *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, Nov. 2015, https://doi.org/10.1109/icdmw.2015.7.