REDDITRANK

Progress Report

Murtaza Latif - 1004307065
Andrew Wang - 1003259305

Total Word Count: 993
Penalty: 0%

Introduction

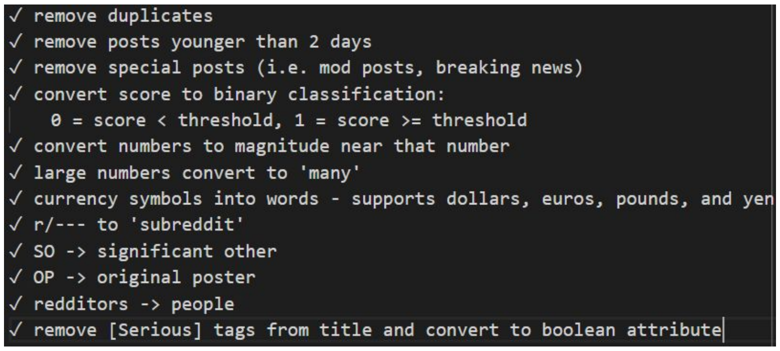
Users can submit posts on Reddit subsites called “subreddits” to be voted by the community. Posts are prioritized by score ("upvotes" minus "downvotes"). Our goal is to predict if "AskReddit" subreddit posts will pass a 100 score threshold. Neural networks are well-suited given their popularity in natural language processing and the labelling intrinsic to posts. Our hope is that industry and users alike can use our predictor to improve advertising or content by providing a means for self-validation.

Data Processing

The PRAW package in Python can fetch a maximum of 1000 posts at a time. Given average 1000-post turnover occurs every 2 hours, our data collection method automatically collects 1000 posts from the AskReddit subforum every 2 hours with the subreddit sorted by ‘new’ (stored as CSV files).

We store only the ids of the newly fetched posts. After 2 days, we extract the information of each of the submissions in our stored ids list to ensure data comes from posts older than 2 days.

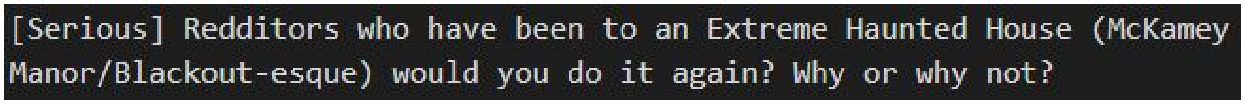
There are several filters we apply to our data, each of which is shown in the following figure:



```
✓ remove duplicates
✓ remove posts younger than 2 days
✓ remove special posts (i.e. mod posts, breaking news)
✓ convert score to binary classification:
    0 = score < threshold, 1 = score >= threshold
✓ convert numbers to magnitude near that number
✓ large numbers convert to 'many'
✓ currency symbols into words - supports dollars, euros, pounds, and yen
✓ r/--- to 'subreddit'
✓ SO -> significant other
✓ OP -> original poster
✓ redditors -> people
✓ remove [Serious] tags from title and convert to boolean attribute
```

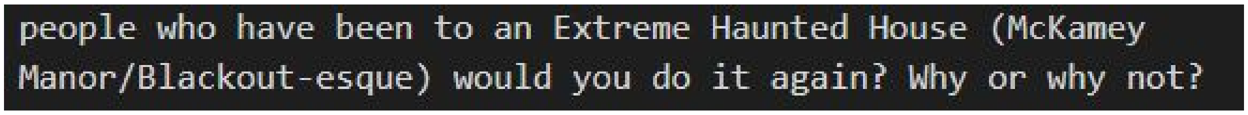
Figure 1: *List of filters applied to our data*

The following figures show an example of a title being put through the filters:



```
[Serious] Redditors who have been to an Extreme Haunted House (McKamey
Manor/Blackout-esque) would you do it again? Why or why not?
```

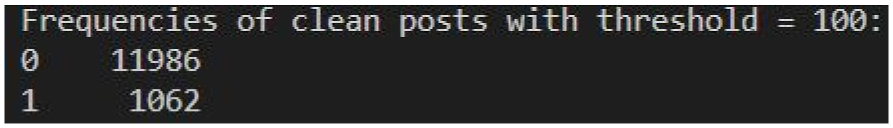
Figure 2: *A title of a post before filters were applied*



```
people who have been to an Extreme Haunted House (McKamey
Manor/Blackout-esque) would you do it again? Why or why not?
```

Figure 3: *A title of the same post after filters were applied*

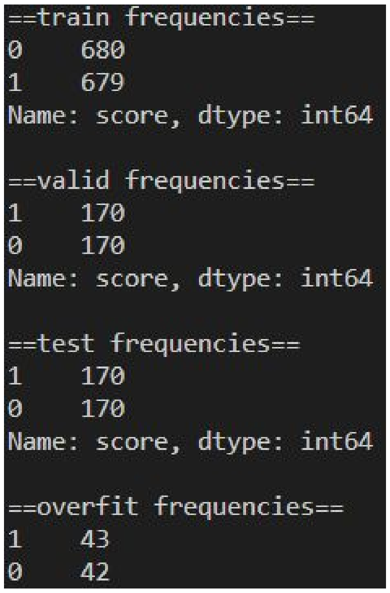
We have collected over 13,000 submissions which are each put through the filter. The posts have a heavy skew towards lower scores as shown below:



```
Frequencies of clean posts with threshold = 100:
0    11986
1     1062
```

Figure 4: *The frequency of scores where 0 represents a score less than 100, and 1 represents a score greater than or equal to 100.*

To obtain better results in training, we balance our dataset within the splitting process:



```
==train frequencies==
0    680
1    679
Name: score, dtype: int64

==valid frequencies==
1    170
0    170
Name: score, dtype: int64

==test frequencies==
1    170
0    170
Name: score, dtype: int64

==overfit frequencies==
1    43
0    42
```

Figure 5: *The frequency of scores for each split dataset where 0 represents a score less than 100, and 1 represents a score greater than or equal to 100.*

The number of samples per class reduced to the minimum number of total samples per class, which results in a large cut of data.

Baseline Model

Originally, the baseline converted titles to frequency vectors to pass into an MLP (Multi-Layer Perceptron). Due to small average title size (10-15 words) and large word vocabularies, this approach yielded sparse vectors with poor training results.

Hence, the baseline now converts title words into 100-dim embeddings (using GloVe) and averages per title. Context booleans representing whether the post is flagged as "serious", "spoiler" or "nsfw" (18+) are additionally concatenated. The resulting "title-with-context" vector is passed through a two layer MLP with 103 neurons and 64 neurons respectively. In conjunction, a "BCEWithLogitsLoss" loss function and "Adam" optimizer are used. A sigmoid activation function is applied after the first MLP layer.

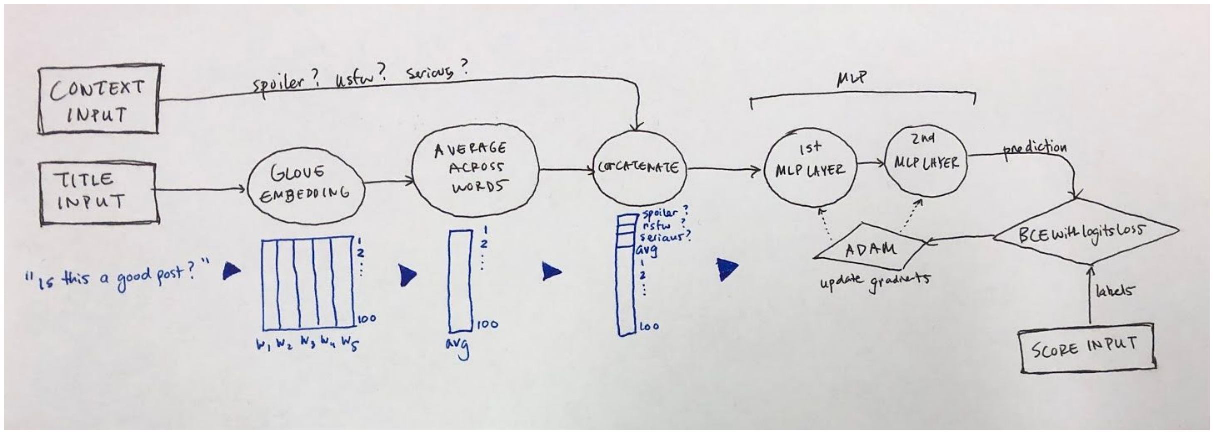


***Figure 6**: Baseline architecture depicting from left to right GloVe embedding, averaging, adding context, MLP, "BCEWithLogitsLoss" loss function and "Adam" optimizer; blue lines indicate walkthrough of example title data.*

Architecture

We are developing a CNN/RNN based model. Both models embed title words into 100-dim word vectors then utilize the "Adam" optimizer with a "BCEWithLogitsLoss" loss function. Both models' outputs are concatenated with the aforementioned context and passed into the MLP as described above.

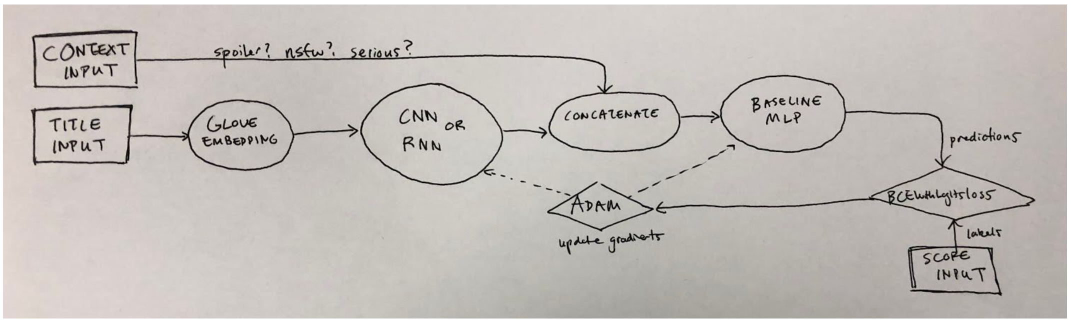


Figure 7: *General model architecture, "CNN or RNN" module to be considered as a "black box" with details expanded on below.*

The CNN passes each set of vectors (per title) into two convolutional layers with 50 kernels sized (100, 2) and (100, 4) respectively. The outputs from these layers are max-pooled along sentence length, then concatenated to yield a 100-dim vector.

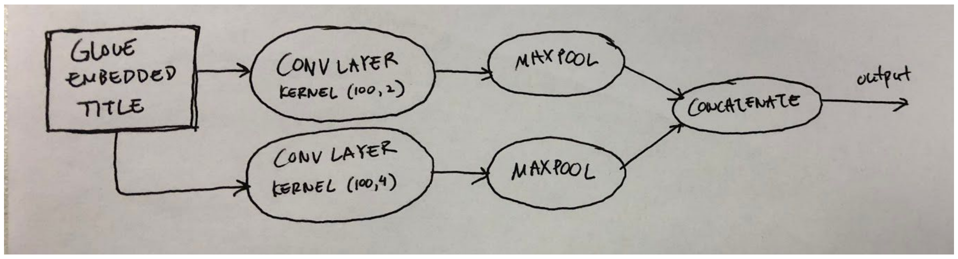


Figure 8: CNN *variant of "black box" in general architecture above*

The RNN passes each set of vectors into a GRU with a hidden dimension of 100. Sentences are "pack padded" using PyTorch's "pack_padded_sequence()" method. The final hidden state is used as its output.

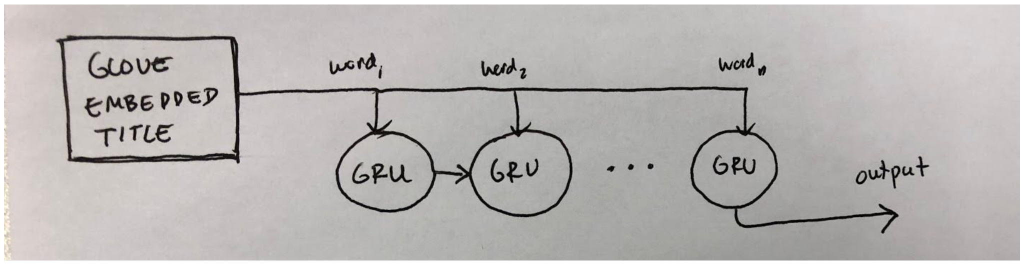


Figure 9: RNN *variant of "black box" in general architecture above*

Result

These graphs of loss and accuracy provide evidence of training (through overfitting). Batch size and learning rates are summarized in the following table:

Table 1: *Batch size and learning rates for the various models*

Model	Batch Size	Learning Rate
Baseline	64	0.1
CNN	64	0.01
RNN	64	0.01

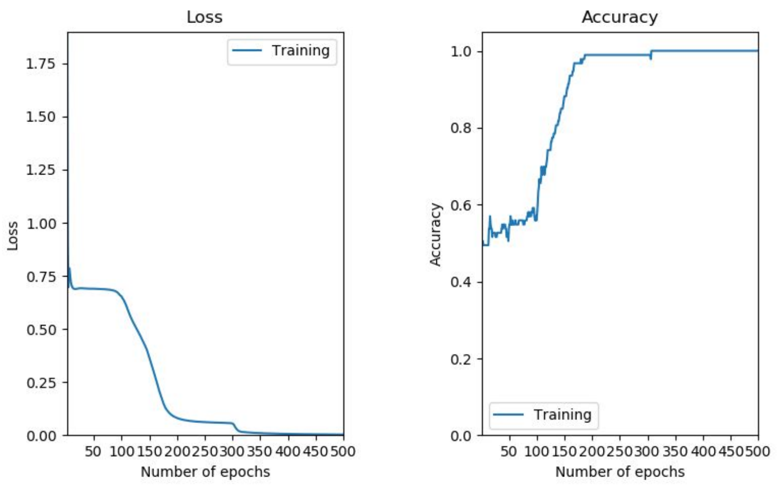


Figure 10: *Loss and accuracy of baseline model over 500 epochs with 40 training examples*

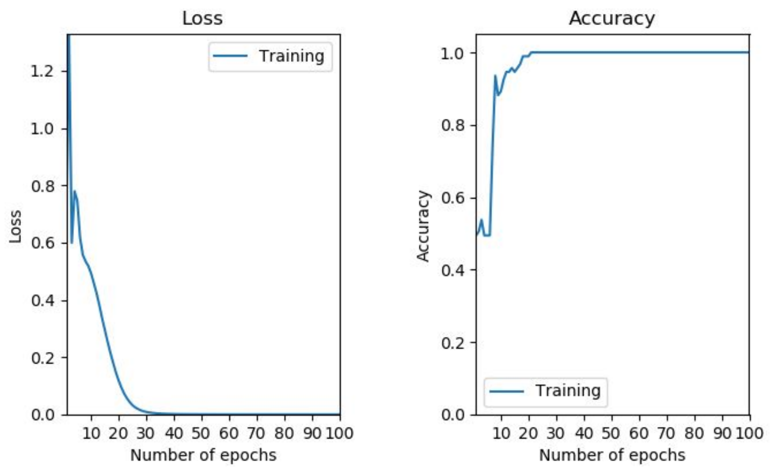


Figure 11: *Loss and accuracy of CNN model over 100 epochs with 40 training examples*

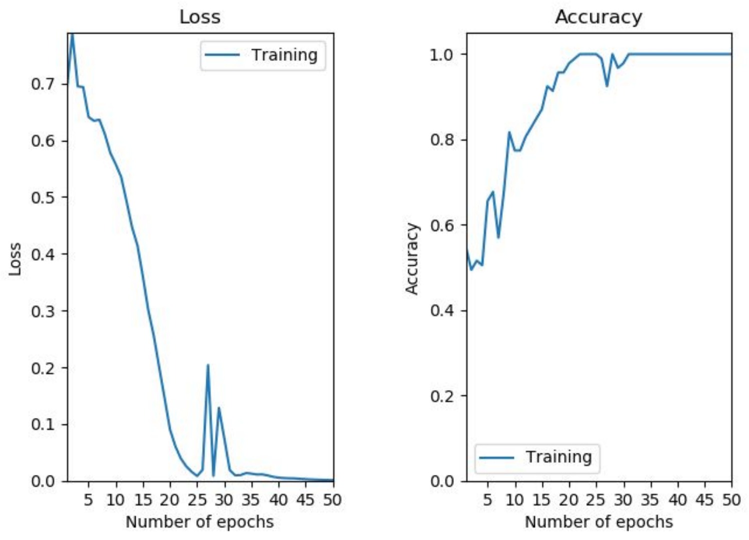


Figure 12: *Loss and accuracy of RNN model over 50 epochs with 40 training examples*

Training/validation loss and accuracy plots are shown below. Breakdown of dataset sizes can be found in **Figure 5**. Results are summarized below.

Table 2: *Summarization of results for each of the models*

Model	**Maximum validation Accuracy (%)**	**Test Accuracy (%)**	**Training Time (s)**
Baseline	57.14	52.70	129.78
CNN	64.15	62.70	123.02
RNN	66.57	62.70	326.95

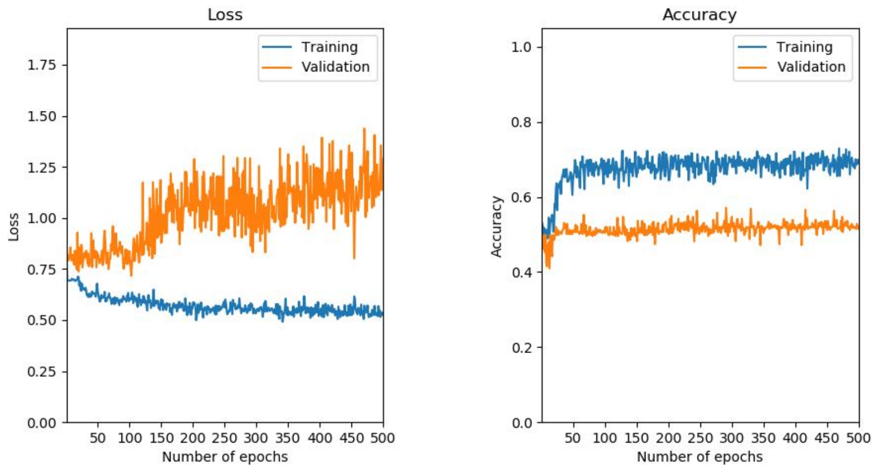


Figure 13: *Loss and accuracy for training and validation sets for the baseline model*

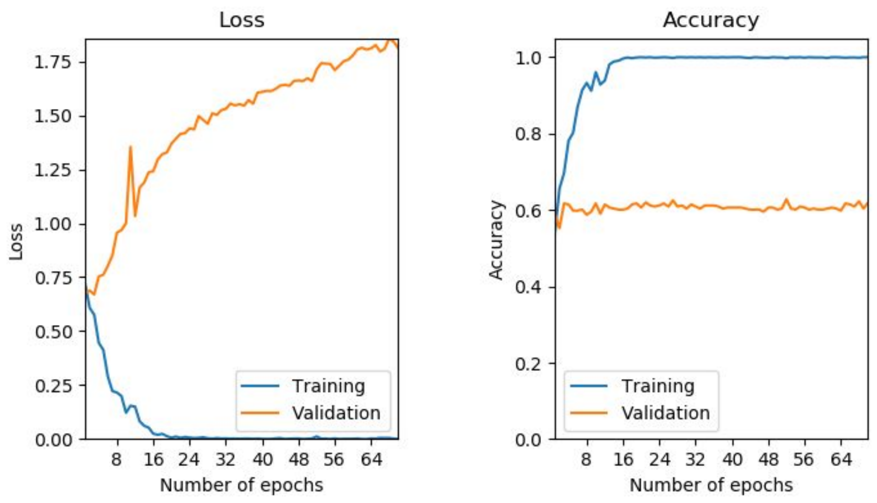


Figure 14: *Loss and accuracy for training and validation sets for the CNN model*

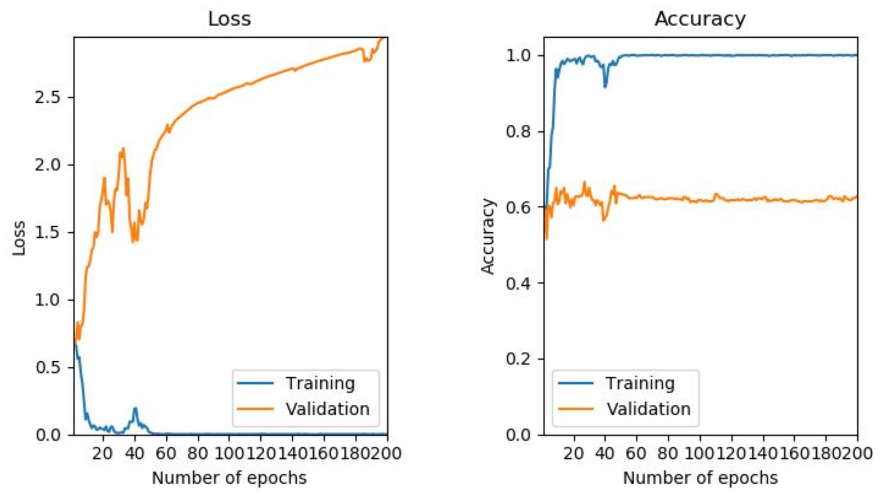


Figure 15: *Loss and accuracy for training and validation sets for the RNN model*

Confusion matrices for these results are summarized in the following table.

Table 3: *Confusion matrices for each of the models*

Baseline	**Predict 0**	**Predict 1**	**CNN**	**Predict 0**	**Predict 1**	**RNN**	**Predict 0**	**Predict 1**
Answer 0	54	131	**Answer 0**	118	67	**Answer 0**	109	76
Answer 1	19	166	**Answer 1**	78	107	**Answer 1**	69	116

Discussion

As evidenced by **Table 2**, the CNN and RNN have better test accuracy than the baseline. The RNN does better than the CNN for validation accuracy, but it takes ~3x longer to train. The graphs indicate that the CNN converges in the fewest number of epochs, followed by the RNN, then the baseline. Looking at **Table 3**, the baseline tends to predict a higher score, with significantly more false positives than the other models.

Worryingly, all three models show increasing validation losses over time, indicating overfitting. This is also seen with the large gaps between training and validation accuracy.

Team Work and Progress

Andrew focused on construction and training of model/architecture. Murtaza focused on data collection and processing. We keep each other updated on tasks and summarized code written to keep a consensus on project direction. So far, Murtaza has completed data collection and processing methods, and a CLI to collect and process data. Andrew has constructed each of the architectures including the baseline, CNN, and the RNN model.

Project Plan

***Table 4**: Task, deadline and responsibility breakdown plan*

Project Task	Responsibility Breakdown	Deadline	Team Member
Data Collection	• Continuously collect data using autocollect • Combine, clean and process data when significant amount of data is collected	Throughout project until 12-02-2019	Murtaza
Performance Analysis of Model	• Run training loops of the models with different datasets and hyperparameters • Obtain statistics and graphs of results to measure improvement of model over time	Throughout project until 12-02-2019	Andrew
Model Enhancement	• Make adjustments to the architecture and hyperparameters of the models based on performance analysis • Keep a record of the best hyperparameters used; save the best models in a file	Throughout project until 12-02-2019	Andrew
Data Expansion	• Identify better ways to implement submission attributes to give more context	Throughout project	Murtaza

	to the neural networks • Analyze datasets to find filters that enhance the representation of the sentence with GloVe word vectors	until 12-02-2019	
Final Presentation	• Collect figures, data and statistics for the presentation • Create presentation slides and content	11-26-2019	Murtaza and Andrew
Final Report	• Collect figures, data and statistics to use during the report • Writing the report (will be further split upon receiving the document outline)	12-02-2019	Murtaza and Andrew