Text-based Data Predictions on “Palworld” Steam Reviews

Kevin Nguyen, CSCI 325

Goal: Label reviews with a topic

- Take new reviews that come in and label them with a general topic

- From the game devs point of view: easier to sift through reviews to find out what is being done well, and what issues are present
    - Steam reviews already have good/bad review indicator, with topics we can tell quickly why someone thought it was good or bad

How will we do this? (Methods)

- Preprocess text (NLP) - prepare text for topic modeling
  - Spell check and English check
  - Remove things like: stop words, punctuation
  - Lemmatization
  - Tokenize text
  - Libraries: textblob, langdetect, nltk
- Topic modeling (LDA) - creating labels for neural network
  - Perform LDA on processed reviews to create topics
  - Label our training data
  - Libraries: gensim
- Neural Network - train a model to correctly label reviews
  - Things to consider: structure of network, concern of overfitting, optimizer, other hyperparameters
  - Evaluate network with testing set
  - Libraries: PyTorch

Findings/Things to note from preprocessing + LDA

- After running spell check, there are some words that are spell checked when they should not be - i.e. “pokemon”
- First pass of lemmatizing and tokenizing text revealed common/obvious words, which we remove
  - “game”, “pal”, “palworld”
  - Side note: some reviews become empty (NaN) - we drop them

- Ran multiple combinations of parameters for LDA
  - Best: 8 topics, 25 passes, take token if in > 100 texts, < 75% of all texts
  - We tried 6, 7 topics - found that sometimes one or two topics were incoherent

My interpretations of words + topics

1. game went above expectations - people not expecting game to be as fun/good as it was
2. talking about different mechanics and design of the game - survival/(open?) world/catching monsters/building
3. talking about how good the game is despite being early access / worth the buy even in early access
4. from a quick a peek at some reviews, there are a lot of jokes about how you basically capture monsters and have them work for you - i.e. slavery, so these are those jokes I think
5. lots of reviews specifically talking about the base building aspect of the game
6. two pokemon games (scarlet/violet and legends arceus) came out around the same time palworld came out - people comparing the two games, saying palworld doing what pokemon did not?
7. reviews talking about the multiplayer aspect - good game with friends, server(issues or stable?), would recommend to friends
8. in general comparing palworld to other games like: pokemon(scarlet/violet and legends arceus), ark survival evolved, minecraft, valheim

Neural Network Structure

- Topography: input - 2 hidden - output
  - Input layer size = 23773 (number of unique words in our texts)
  - Hidden 1 = 64, Hidden 2 = 32
  - Output = 8 (number of topics)

- To help with overfitting issues, we employ dropout with rate 0.2 between Hidden 1 - Hidden 2 and Hidden 2 - Output

- We use ReLU (rectified Linear Unit) as our activation function
- We use cross entropy loss as loss function - good for multiclass classification

Training the Neural Network

- We run the Adam optimizer, with a learning rate of 0.0025 and weight decay (L2 regularization) of 0.005
- Number of epochs = 30
- Batch size = 32

- We keep track of loss, accuracy, recall, and precision for training and validation set
- We plot the training loss against the validation loss to evaluate our model

Overfitting Issues

- Could have been that I just did not have enough data.

- Tried many combinations of:
  - learning rate (0.001, 0.0025, 0.005, 0.0075, 0.01, 0.05, 0.025, 0.03)
  - Batch size between 10 - 50
  - Either overfitting or just not converging at all
- Tried adding 3rd layer - realized this is counterproductive
  - Giving the model more time/more of a chance to overfit on training data

Overfitting Issues (cont.)

- Needed to try something else

- Incorporate dropout and L2 regularization
    - Using the base learning rate 0.001 and batch size 32
    - Tried dropout between hidden layers and output with rate 20%, 15%
    - and weight decay of 0.0001, 0.001, 0.005
- Reduce neurons in layers to reduce number of parameters
    - Went from (128, 64) to (64, 32)

- Much better results!

Performance Metrics of Model

Values taken from 30th epoch

Training Set

- Loss: 0.4836
- Accuracy: 0.8512
- Recall: 0.7625
- Precision: 0.8455

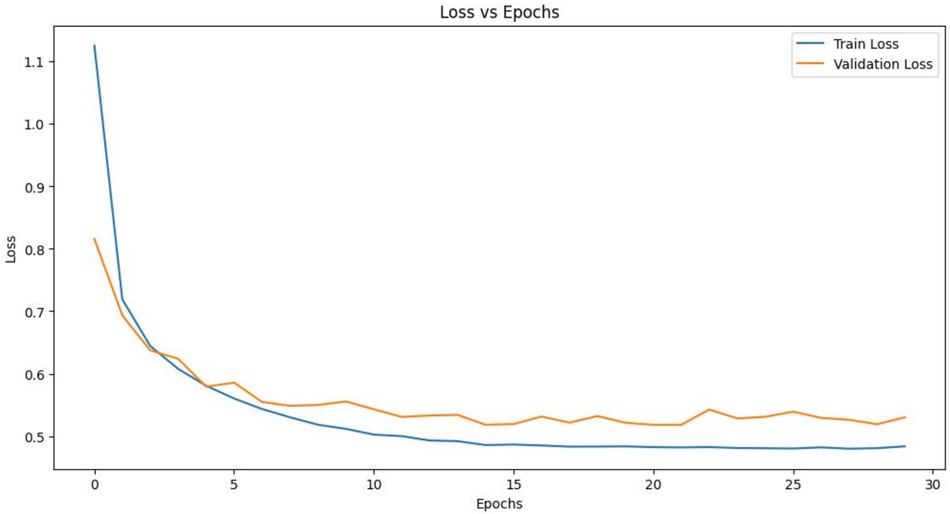
Validation Set

- Loss: 0.5300
- Accuracy: 0.8320
- Recall: 0.7500
- Precision: 0.7931

- It is worth noting that these values were oscillating slightly (see notebook for all epochs)
- So consider these values with a small error, like ±0.02, i.e. close to 0.5 for loss and 0.8 for Acc., Rec., Prec.

Analysis of Metrics

- We get convergence to around 0.5
- Considering in some tests, loss went up to 4 (and was increasing), 0.5 seems okay?
- Also considering this is one of the few configurations where we converge
- Difference between train. and val. loss is relatively small
- Accuracy, recall, and precision values are decent - ~0.8



Trying model on the testing set

- After applying our model to the testing set, we take samples of size 10 and subjectively check if topics look accurate.
  - Repeat 5 times (50 samples total)
- From the first 4 samples
  - 23 good, 9 I deemed hard to tell / not good or bad, and 8 were bad
  - If we are optimistic and group 23 + 9 together, we get accuracy of about 0.8
  - If we are critical and group 9 + 8 together, we get accuracy of about 0.575
  - See notebook for more details on criteria as well as 5th sample
- Is this good? Probably not. We most likely converged to a bad local minimum.

Conclusions + Future Work

- Our model has somewhere between 60%-80% accuracy on the testing set - worth noting I am biased
  - i.e. our model is not very good
- Might consider redoing some of the preprocessing to ensure words like “pokemon” are kept as is, or do more digging to remove common and not useful words
- I think our choices in LDA were okay, but still could experiment there
- Might need to tune the neural network more to get better convergence
- Might just need more data!