

## **Generating Game Reviews**

Michael Horst, Jordan Timmerman, Agam Gupta, Upasna Madhok

Contact: [MichaelHorst2016@u.northwestern.edu](mailto:MichaelHorst2016@u.northwestern.edu)

EECS 349: Machine Learning  
Northwestern University, Spring 2016

### **Motivation**

Steam (an online gaming platform that provides users with installation and automatic updating of games in addition to many other features) has a database of games that have over tens of thousands of reviews, not all of which are high quality/useful and a lot of which are similar to one another. We think generating reviews that can accurately emulate the best reviews on Steam will give prospective buyers more information, especially if the model can generate high-quality reviews of a game based upon a data set of lower quality reviews. This will save potential customers time in evaluating whether a game is worth purchasing.

### **Solution**

We focused on two different machine learning solutions to generate helpful reviews: character-level Recurrent Neural Networks (char-RNN) and Hidden Markov Models (HMM). char-RNNs train on a chunk of text and learn to generate new text character by character, whereas HMMs calculate transition probabilities between n-grams (sequences of n words) for training, and then can be sampled to generate text. To train char-RNN, we used text files containing review content ranging from 300 kB to 15 mB, epochs ranging from 10-50 (more for smaller data sets) and hidden units ranging from 48-128 (more for larger data sets). We then generated samples with temperatures ranging from 0.3 to 0.6. For HMMs, we wrote generic model generation and sampling scripts and trained multiple models. First, we tried taking the top 50 most helpful reviews for a specific game and creating bigrams and trigrams. Then, we tried taking a sample of 15000 out of the set of top 50 most helpful reviews from every game in the data set of 8145 games, and created models with bigrams, trigrams, all the way up to 10-grams, and tested the output. 10-gram models generated almost verbatim copies of very long reviews from the data set. 3-5-gram models generated coherent, but sometimes confused, reviews that were fairly original but which might talk about one game one paragraph, and another game in the next - and mention each by name. Results were better when training data was limited to a set of reviews for a particular game, although the results were often passable even with the large training set.

### **Training**

Our data was extracted from the Steam API. We first obtained a list of all games with their appID (the game ID), review score, review sentiment and reviewcount. 8145 games were scraped. We then iterated over all these IDs, calling an endpoint for 'homeContent' from which could be accessed HTML reviews, which were then parsed and processed to generate a

dataset for every review of every game containing *reviewerUsername*, *reviewerDisplayName*, *reviewerProductsCount*, *reviewText*, *recommended*, *hoursPlayed*, *commentsCount*. For each game, the reviewText columns of the review CSVs contains the actual text content of reviews by different users (with newlines escaped; our RNN and HMM then unescape newlines during generation).

For Char-RNN, data was preprocessed using *pandas* and other useful libraries. We would store selected reviews, weigh them based on different approaches, and then dump them to a text file, which is then used to train the CharRNN model. This was done for every game in the database.

For the HMM we use a slightly different approach. Bash scripts using GNU Parallel, awk, and other unix utilities were written for slicing and dicing the review csvs. These scripts were then used to take the top 10, 25, and 50 most helpful reviews from every game and dump those into their own csv files. The HMM was trained using one of those data sets, and then the model was dumped to a python pickle file. A separate HMM sampling script was then run to generate reviews from the model.

## **Methods and Models**

We used these machine learning techniques because char-RNN and HMM are unsupervised learning algorithms that generate statistical models which can then be sampled to create psuedo-original text. Unsupervised learning was necessary because the variables going into the creation of a review were unknown, and statistical models provide a convenient sampling source for text generation.

### **Char RNN Model**

We tried different ways of creating the training text file since that is used for training the model. We used a couple of different approaches with varying values of temperature, number of epochs and number of hidden units so as to generate the least possible perplexity.

The various approaches used were as follows:

#### Approach 1: Use the top 1000 reviews of a particular game as the training set

In this scenario, we took the top 1000 reviews of a particular game (such as CounterStrike or Dota - each having over a million reviews) and stored them into a text file. This was used to train the RNN model.

#### Approach 2: Weighting the reviews

In this approach, we use a weighting function to give useful reviews a higher priority in the training model. Since some of the reviews are gibberish or not useful this approach should help in generating more meaningful text. The following weight function was used:

$$review\_textFile = \sum_{i=0}^{n-1} (n - i) * (reviews\_numbered\ 50 * i\ to\ (50 * i + 50))$$

### Approach 3: Combining the top 3 reviews from all the games

To have a considerable amount of data to train on, we took the top 3 reviews from each game title and stored them all in one text file. Even this amounted to about 12 MB of data, a formidable amount on which to train. We had initially considered the top 50, but that amounted to 160 MB, not feasible for the computing power we have available. To demarcate a particular game, we added the name of the game prior to every review corresponding to that game in the text file. This was done so that we have adequate information to train on and when we use the model to generate a sample, we can potentially ask it to generate reviews for a particular game based on keyword search.

## **Hidden Markov Model**

We created word-level Hidden Markov Models, as opposed to character level (like Char-RNN) both for the purposes of contrasting the two approaches, and because the contextual nature of n-grams lends itself fairly well to sentence generation from probabilistic sequences of words. We experimented with varying values of “n” for our n-grams, from 2-10, on sets of up to 15,000 reviews taken from the set of 279,520 top 50 most helpful reviews sampled from 8,145 games. We found that unigram-generated reviews were largely unintelligible, and that reviews generated with 5-gram and above became borderline plagiaristic. Bigram models worked well, as did trigram models, but 4-gram performed best overall. Larger values of n were less likely to produce short reviews (which makes sense), and so using a variety of values of n helps to generate the most realistic set of reviews in total. We were able to generate game-specific reviews by taking data only from a specific game (given that enough data was available), and these tended to be pretty good - see the following reviews generate for Call of Duty: World at War (there are 4 examples; all 4-gram):

"No hackers in my history teacher."

"BANG BANG BANG BANG BANG BANG !CLING! M1Garand 11/10"

"Totally it just for zombies alone. the campaign is decent, but the last mission (veteran difficulty) is ridiculous, how did the german army lose if they had that many grenades? Modding is keeping this game alive ever since its release back in 2008, my personal favourites is the 2nd best Call of Duty: World at War's zombie mode that was unlocked upon finishing the game. During this time I didn't have internet so I am only getting half of the best cop-op experiences I have ever had. Its fun."

"Its one of those games that are so gorey that it makes you appreciate the army's work, for example you shoot someone in the day..."

Figure 1: examples of reviews generated for a specific game

We also generated reviews on a cross-sectional data set of 15,000 of the 279,520 top 50 most helpful reviews from all 8,145 games, and here are two examples (both generated from a 4-gram model):

"Its combat is more like an advanced version from Disciples 2. The Combat area is smaller now, and every unit has their own limited movement and attack range, which makes formation extremely important. Ships and thief abilities are back, too! How could a old Disciples Fans miss all these features.

Although there are still a few balance issues (poor Alliance QAQ...), and a little bugs as well (we have some non-official fix. don't know whether the steam version has fixed them or not), the game has the whole mechanic remastered and fixed all the buggy and annoying problems from the old 2.

Moreover, Reincarnation is very mod-friendly. It provides map editor and file-export function. Players could make mod themselves in a very convenient way. Don't miss it! (Really hope it could provide Workshop support

Although, not sure if I missed a game or something but I dont know who he is nor do I give a toss about him.

The auto save function doesn't overwrite itself so with each quicksave you will create a new save file.

The characters wont shut the hell up. The talk whenever you click anything and the voice action is alright at best.

Do not buy this game. Its just not worth it."

"This is not too surprising when you consider that the studio that made it, Darkworks', previous horror game before this was Alone in the Dark: New Nightmare, another Resident Evil-clone game. In fact, the studio seems to have this big thing for horror, as they have released six games in the last 15 years, and half of them are horror titles (the newest one they released was I Am Alive).

Cold Fear plays like possibly the truest form of new Resident Evil meets old Resident Evil I have seen. The game switches up between a Resident Evil 4-style camera (when you're aiming, during certain gameplay segments), and fixed camera angles (ala' old-school Resident Evil). Assuming you aren't horrible with fixed-camera angle controls, the transitions were done pretty well and weren't nearly as disorienting as I thought they might be. They were handled and placed well in my opinion. A few good subtle scares, mixed with shoot-shoot, bang-bang, but with some limited ammo edge (though not extensively). And while it isn't the prettiest game around (even back for when it came out it was a dissapointment.

Last I checked this game was non-functioning as its main focus was online and no one played it because it sucks but mostly because the servers were down and were down for the foreseeable future. Im not sure as to whether they are online now but I doubt the game has changed since I last played it so unlike other games I don't really feel the bother to install and try it. The dev blog or whatever's out there should be able to tell you if interested.

Stay away from this one is about the best thing you could do."

*Figures 2 & 3: some of the better reviews generated from a 4-gram model*

Here's an example 4-gram model output that shows cross-game confusion (mentioning both SPAZ and Batman: Arkham Asylum) and bullet-list generation ('+'s are "pro" and '-'s are "con"; this is a known convention in the Steam community):



"I prefer face-to-face conversations. In case the situation turns bad, I know exactly who needs to be blown up."

Space Pirates and Zombies (SPAZ) is a hybrid 2-D space simulation crossed with a real-time strategy game developed by the company Minmax Games with emphasis on combat mechanics and tactics.

Atmosphere & presentation:

- + Large amount of ships with creative designs guarantee a lot of variety and enable a corresponding learning curve.

- + The salvage and mining mechanics keep up the motivation to explore new star systems.

- + Size and mass of ships influence movement and inertia of each vehicle.

- + Late game introduces an ever shifting equilibrium of power.

- o The faction system adds an interesting dimension to the game, but it has little impact on the behaviour or decision making of the player.

- Amount of side missions and you have your own hegemony/league?).

Also, the game allows you to play differend kinds of factions with differend kinds of gameplay (for example, the Paeonians are mainly a cavalry faction with no hoplites, the Illyrians got no cavalry while having a huge empire so they have to focus on the mission objective and you can solve any way you want to get the job done. Maybe this sentence as an example will be a bit spoiler so I warned you: the mission is to destroy 2 battery guns, you have 4 soldiers against lot of enemies, you can easily sneak to the first battery gun and instead you blow it you can use to take out the enemies using stealth.

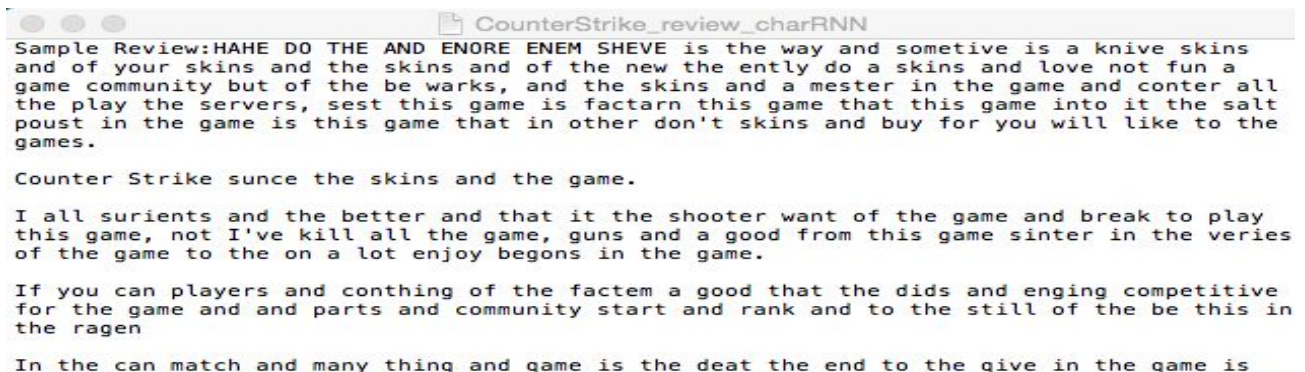
Almost all of the major villains are back from Arkham Asylum, proving Batman should really start killing his greatest foes. Not just that, but there are new villains and almost all of them have their own gang. This gives the goons more style, but really, the goons are all the same for the most part resorting to just brawlers and gunmen. Eventually the game amps things up with long range snipers, armored brawlers, shielded brawlers, knife wielders and stun batoners. If a gun or weapon gets knocked out of the hand of a gunman, someone else can pick it up. The brawlers will pile on you, sometimes two dozen at a time, and as Batman, you can easily if not tediously defeat them. Batman: Arkham City is essentially a beat em up with a very good story.

*Figure 4: 4-gram model generated text with bullet list and cross-game confusion*

## Results & Analysis

After implementing the HMM and trying different approaches for char RNN we realized that HMM performs much better at generating sensible and useful results than RNN.

The best result we got for char RNN on an individual game was through approach 2, ie., by weighing the inputs, and that led to a perplexity of 7.959. We learnt that decreasing the value of the hidden units gives lower perplexity as the model does not overfit. Additionally, in char RNN the perplexity goes down exponentially, decreasing extremely fast for the first few epochs and then mostly being stable until it rises again due to overfitting. Thus through multiple tests, a suitable value of 48 hidden units, and 50 epochs was chosen to get the best possible results.



CounterStrike\_review\_charRNN

Sample Review:HAHE DO THE AND ENORE ENEM SHEVE is the way and sometime is a knife skins and of your skins and the skins and of the new the ently do a skins and love not fun a game community but of the be warks, and the skins and a mester in the game and conter all the play the servers, sest this game is factarn this game that this game into it the salt poust in the game is this game that in other don't skins and buy for you will like to the games.

Counter Strike sunce the skins and the game.

I all surients and the better and that it the shooter want of the game and break to play this game, not I've kill all the game, guns and a good from this game sinter in the veries of the game to the on a lot enjoy begons in the game.

If you can players and conthing of the factem a good that the dids and enging competitive for the game and and parts and community start and rank and to the still of the be this in the ragen

In the can match and many thing and game is the deat the end to the give in the game is

*Figure 5: The review summary generated by Char RNN for a temperature of 0.5, hidden units = 48, number of layers = 2*

As seen through figure 3, the generated summarized review doesn't make much sense, and so we could not extract its sentiment to compare with the average review sentiment from the initial test data.

The results from the HMM, trained either on specific game reviews or cross-sections of the most helpful reviews across all games, were much more productive. The summarized text that we got mirrored the general language structure of a game's review fairly closely - sometimes even mimicking bullet-lists of pros and cons. Since large values of  $n$  for our  $n$ -grams tended to be fairly unoriginal and plagiaristic, and since small values of 1-3 tended to produce short reviews that were sometimes unintelligible (correct grammar, but no meaning), 4-grams proved to be the most effective.

#### *Brief suggestions for future work.*

Our most successful Hidden Markov Models used an aggregate of the top 50 reviews from 8,145 games, and while these tended to look like fairly "real" reviews they are hard to associate with any particular game. We took the top 50 most helpful reviews because review quality tends to drop off steeply after about that value. Models on specific games tended to perform worse, largely because many games did not have enough high-quality reviews for the model to be effective. (Only 3,332 of the 8,145 games had more than 100 reviews total). Future work could improve upon the more general model by seeding it with the name of a game; providing transition models for common review components such as ratings (eg "11/10"), pro/con lists, and section headers; or adjusting the weights of the model to favor less common review elements more and discourage repetitive content.

#### *Who worked on what*

Agam, Upasna and Jordan worked on scraping review data from the API. Michael and Agam compiled the text documents of varying review sets and trained the char-RNNs. Jordan created a flexible script to create various review sets in .csv format for HMM training, and programmed the HMM algorithm itself. Everybody worked on the report.

Link to GitHub: <https://github.com/skorlir/349-project>