Classification and Topic Modelling on Video Game Reviews

Sumeet Dubey
Under Professor David Smith
Northeastern University

Table of Contents

Abstract

Writing reviews is a popular form of communication to assess or express views on a variety of things. As the sentiments of the writer are conveyed through the words they use, these words give us an intuition about identifying these sentiments, and in-turn allow us to build models that can learn and infer from them. In the context of this paper, the focus will be on videogame reviews. I have picked a dataset of written game reviews accompanied with a score, and performed text based classification and topic modelling on it. I use variants of Naïve Bayes Classifier and compare their performances. I then run two topic modelling algorithms to identify games similar to a given game, based on topics believed to be present in it's review text. Towards the end, performance of all these models is discussed along with some potential future work.

Motivation

I decided to work on this dataset because it allows to implement some interesting text based models. Video game industry has been around only for a few decades that makes it fairly new compared to many other industries. It is also a vastly growing industry with multiple consoles releasing every year, that in turn provide new games. Written reviews (along with video reviews) is one of the most used methods for giving scores to games. It is therefore interesting to analyze as to what extent these reviews help us in identifying the writer's sentiments and predict scores based on the review text for evaluating our model.

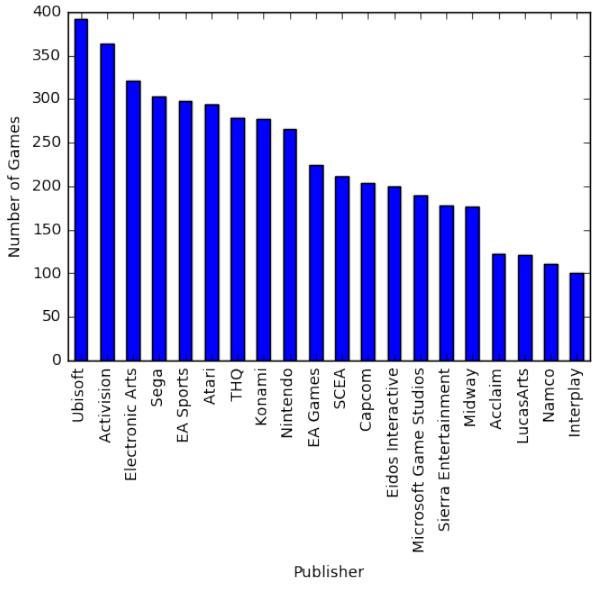
Dataset

In this dataset, we have games along with their written reviews by game critics from Gamespot.com, for about 8200 games. Each written review also has a score and some other data like publisher of the game, genre, year of release, etc. The dataset also consists of reviews submitted by users (if any), but the focus of this paper will be the Gamespot review. All reviews are text files and are organized into folders by consoles.

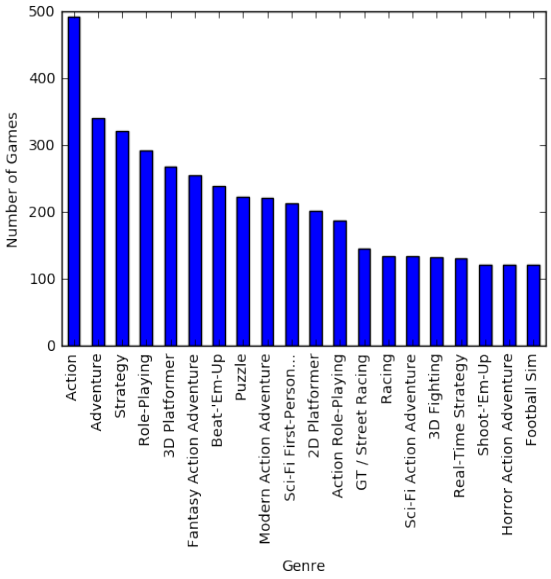
Exploratory Analysis

Top Companies and Genres

Our dataset provides us a fairly good distribution of games from different publishing companies. Noting from the plots below, we can see that all of these top publishers have published at least 100 games each, with Ubisoft, Activision and Electronic Arts encompassing more than 1000 games. It should also be noted that EA Sports and EA Games are both brands of Electronic Arts, implying EA has the highest number of games in this dataset.



Plot 1: Top 20 publishers by number of games



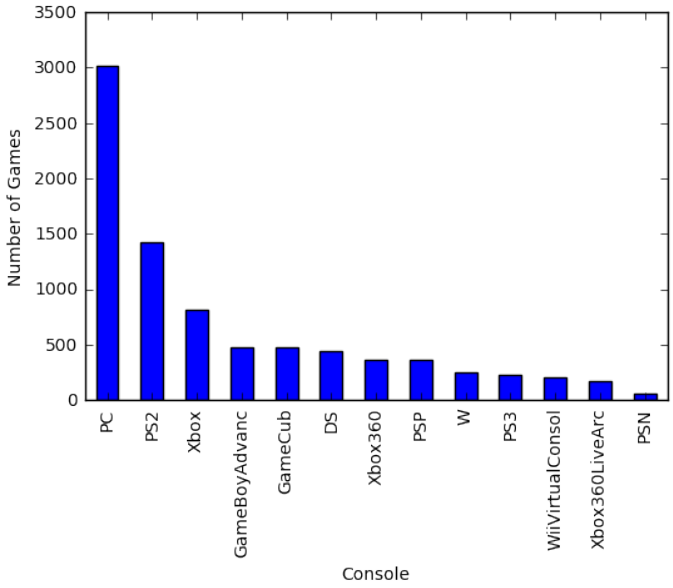
Plot 2: Top 20 genres by number of games

The top 20 genres cover most of the popular genres in games. Few of the top genres such as Action, Adventure, Role-playing are expected as there are a good number of games that contain elements that qualify for these genres. A game that has a level structure and requires a player to finish them one after another, qualifies for an action game. A game based on a character playing out a role in some story can be called a role playing game.

We should also notice that genres in this dataset are rather granular, coming to a total of 137 genres. This granularity can be observed by looking at genres such as Fantasy Action Adventure, considering the fact that all 3 of those words are subgenres of their own.

Number of Reviews by Console

The plot for number of game reviews for each console appears rather skewed. Part of this skew-ness is due to the history of video games, when initially PC was the only platform for which games were released, before the era of consoles. We should except the PC game reviews creating a bigger influence on our models by contributing a lot more features than other consoles.



Plot 3: Number of game reviews for each console

Algorithmic Analysis

For this dataset, we require models that can infer from words across documents. The algorithms we will discuss pertain to classification and topic modelling. We will visit three variants of Naïve Bayes classifiers (Multinomial, Gaussian and Bernoulli) and compare their results using Root Mean Squared Error. We will then apply Latent Sematic Analysis and Latent Dirichlet Allocation to find popular topics, and use their results to find similar games based on these topics.

Preprocessing

The data set is already pretty clean. I converted all the text into lower case and transformed words with punctuations as single words (you're become youre). Game scores were converted to floats before storing. For the Naïve Bayes classifiers, I used 'stemming' to get better features. Stemming is a technique by which different morphological forms of a word are converted into a base word (eg. Laugh, laughable, laughter, laughing could be represented as a single feature laugh). I have used the NTLK snowball stemmer library [1] for the purposes of this project.

Naïve Bayes Classifiers

Naïve Bayes classifiers is a set of classification models that work on Bayes theorem and an independence assumption of features with respect to each other. In other words, if we have n different features as $\{x_1, x_2, ...,x_n\}$ and we wish to calculate the probability that one of them belongs to a class C_k, this probability will only depend on that one feature.

$$p(x_i \mid x_{i+1}, \dots, x_n, C_k) = p(x_i \mid C_k)$$ [2]

The above result when applied in conjunction with Bayes theorem, give us a joint probability distribution across all features as

$$p(C_k \mid x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Z here is known as the evidence of a feature x and is the probability of x over the corpus.

From the above result, we can calculate the posterior probability of every feature belonging to a class. For building a predictive model, we then simply choose the class with the maximum probability for a given set of points.

$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

This naïve assumption of feature independence works really well in many text classification applications. Also, these classifiers are fast in terms of training time.

The variants of Naïve Bayes used differ slightly from each other. What is common is that all these models take some sort of word vectors as inputs. These vectors are simply the word counts of all words in our vocabulary, across all documents. The multinomial NB classifier takes input as word count vectors across the document, whereas the Bernoulli classifier takes a Boolean version of it, where entry K_{ij} in the word matrix will be 1 if word j occurs in document i, 0 otherwise. The Bernoulli classifier takes into consideration only the occurrence of a word rather than its count in a document. The Gaussian classifier takes in a continuous variable x and computes the mean and variance of x from each class. This continuous variable is assumed to be distributed according to a Gaussian. The values of mean and variance are then used to measure the probability of a new data-point to belong to one of the classes.

Topic Modeling

Topic modeling algorithms try to find commonly occurring topics in a corpus. A topic here means a collection of words that often occur together in context, suggesting a relation between them. I have used the scikit learn libraries for LSA [3] and LDA [4] algorithms.

LSA

Latent Semantic Analysis is a technique to generate concepts from a matrix of documents and terms, where the generated concepts help to analyze the relationship between these documents. LSA uses a mathematical technique called Singular Value Decomposition (SVD) to reduce the dimensionality of a matrix and returns three matrices U, S & V^T.

Suppose we have a word-document matrix X, LSA uses SVD on X to give

$X = U S V^T$ (approx.)

where X is a vxd word-document matrix, U is vxk word-concept matrix, S is kxk concept matrix & V is dxk document-concept matrix. Here v is vocabulary size or number of terms in the given dataset and d is the total number of documents.

Algorithm:

1. Read all the text data and store it
2. Perform TFIDF on the output of step 1 by passing it to *Tfidfvectorizer function*. Stop words removal can be done by passing a list of stop words along. This will return a (document, word) matrix identifying word frequencies across documents.
3. Now pass the tfidf matrix to the function that uses *TruncatedSVD* to perform SVD.

After calling the fit and transform methods, the algorithm returns a dxk matrix where d is number of documents and k is the number of topics learnt.

LDA

Latent Dirichlet Allocation is a generative probabilistic model for discrete data such as text. LDA basically considers a document to be a mixture of hidden (latent) topics, where each topic is characterized by a distribution over words. LDA is identical to probabilistic latent semantic analysis(pLSA), except that LDA topic distribution is assumed to have a sparse Dirichlet prior.

LDA uses following generative process:

1. Choose $\Theta_i \sim \text{Dir}(\alpha)$, where $i \in \{1\ldots, M\}$ and Dir(α) is a Dirichlet distribution with symmetric parameter α which typically is sparse ($\alpha < 1$).
2. Choose $\varphi_k \sim \text{Dir}(\beta)$, where $k \in \{1\ldots, K\}$ and β is typically sparse
3. For each of the words in position i,j where $j \in \{1\ldots, N_i\}$ and $I \in \{1\ldots, M\}$.
    a) Choose topic $z_{i,j} \sim \text{Multinomial}(\Theta_i)$
    b) Choose topic $w_{i,j} \sim \text{Multinomial}(\varphi\, z_{i,j})$

where α is the parameter of the Dirichlet prior on the per-document topic distributions, β is the parameter of the Dirichlet prior on the per-topic word distribution, Θ_i is the topic distribution for document m, φ_k is the word distribution for topic k, $z_{i,j}$ is the topic for the j-th word in document i, and $w_{i,j}$ is the specific word[5].

The implementation of LDA is almost the same as LSA in scikit. One key difference is that word vectors are passed as input in LDA in stead of a tf-idf matrix as in the case of LSA.

Results

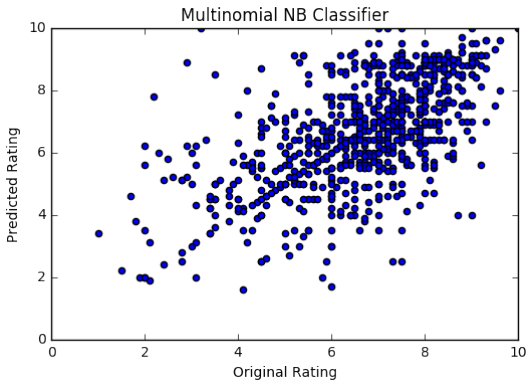
Text classification

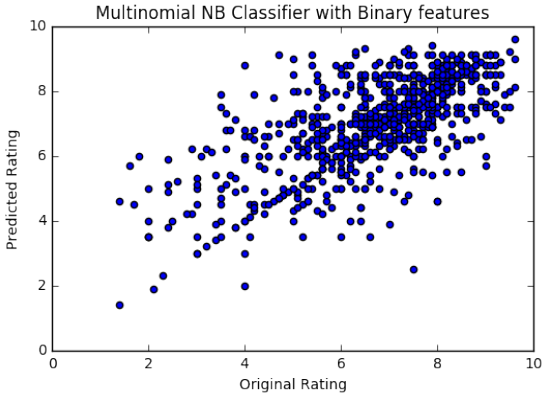
As mentioned above, I have used 3 variants of Naïve Bayes classifiers on our dataset. I divided the dataset randomly into 90% train and 10% test parts. After training the models on the dataset, I compared the Root Mean Squared Error values for their predictions on the test set. I then tested the models again, after applying stemming and the results were better for all 3 models. Out of the 3 models, Multinomial NB model has performed the best. It is interesting to note that binary vectors have performed better that normal word vectors for Multinomial NB model. This maybe hints us that in our case, multiple occurrences of words do not contribute as better features compared to words that don't occur that often. In review text, just the presence of some word might add enough meaning to the sentiments of the reviewer (broken gameplay, bad graphics), irrespective of how many times it occurs.

Table 1: RMSE scores for different models

	Multinomial with word vectors	**Multinomial with binary vectors**	**Gaussian**	**Bernoulli**
Without Stemming	5.080232892832968	3.8827084231837303	8.866872788163063	5.555528705599885
With Stemming	4.516292295181921	2.366729556261516	8.608625306454057	6.105852937203903

Word stemming seems to do a good job in reducing the error scores for all our models. The plots below show us the performance of our models with stemming. Notice the diagonal (y=x) along which the many points are concentrated in our graphs. The points on this line are games for which the model was able to do a prefect score prediction. Amount of deviation from this line gives us an intuition about the error in the model.





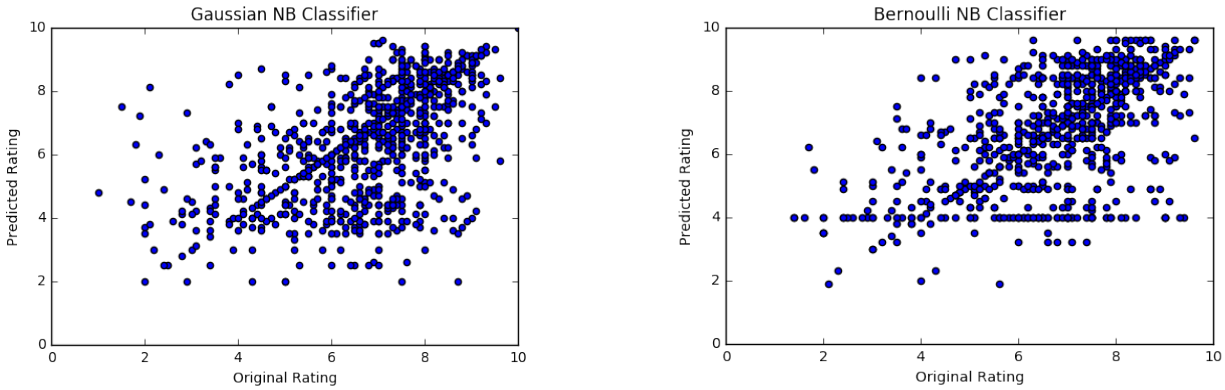


Figure 1: Plots showing variance in predicted and original ratings for different models

Topic Modelling

Both topic modelling algorithms performed well in identifying common topics amongst games. I first ran LDA on four subsets of the dataset to get a general sense of the topics present in our corpus.

On entire dataset: LDA does a good job in identifying some common topics used in game reviews as we can see from the results. Most of these topics adhere to a particular genre/console, giving an intuition about the documents that contain those topics.

On top publisher titles: Results seen here are similar to above, having dedicated topics to some genres and consoles.

On titles having GameSpot Rating >= 8: Words and topics returned for these games have a rather positive sentiment. This can be noted by observing words such as new, good, great and different. This is expected as these reviews have more positive words in them corresponding to their high rating.

On titles having GameSpot Rating <= 4: As we might have expected, the topics here seem to have a negative kind of sentiments. This is evident by spotting words such as bad, broken and problems.

```
Number of inputs: 2067

Topic 1:
combat campaign strategy battle enemy missions multiplayer battles mission different
Topic 2:
version pc xbox youre missions good versions great action need
Topic 3:
characters story character youre make different good great way theres
Topic 4:
players player mode years new team year control make right
Topic 5:
players youre mode online team player teams make different sports
Topic 6:
new original make players youre features theres great better world
Topic 7:
mode new really series good years way great youre sports
Topic 8:
characters character players fighting mode team online different skills gameplay
Topic 9:
mode series new youre different tracks speed track online way
Topic 10:
action weapons enemies missions youre good way use different multiplayer
```

```
Number of inputs: 590

Topic 1:
bad play theres action enemy youll pretty characters character look
Topic 2:
youll enemies combat characters action attack time youre level levels
Topic 3:
youre theres way actually going time big little new things
Topic 4:
mode play youll time youre button fun players gameplay different
Topic 5:
levels world mode time level youll gameplay make sound play
Topic 6:
level isnt gameplay fun little youll way really thats play
Topic 7:
youll time play theres players youre thats way bad look
Topic 8:
missions mission youre enemy way combat enemies isnt time make
Topic 9:
good youre make doesnt dont sound lot youll need hard
Topic 10:
version original screen graphics new youll little use need make
```

Figure 2: Topics for games rated above 8 and below 4

LSA and LDA on game similarity

To do a comparative study on the topic modeling algorithms, I have tried to observe how the topics in our models relate to the actual documents. To do this, I calculated a similarity score across all documents in our corpus. The measure I have used to calculate document similarity is cosine similarity. Cosine similarity computes similarity as the normalized dot product of matrix X and Y as:

$$K(X, Y) = \langle X, Y \rangle / (||X|| * ||Y||)$$ [6]

In our case, we pass the outputs of our models (n_documents x n_topics matrix) to the cosine function as both X and Y. This gives us two (n_document x n_document) matrices where each entry S_{ij} is the similarity score between documents i and j in our two models. If S_{ij} is high, we can conclude that document i and j are very similar, according to the selected model. A manual comparison of top 5 similar games for 5 individual games is done below. I have tried to choose games with varying features (in terms of content, publisher, genre and console) to give us a good sense of the models.

Game 1: Pokemon Leafgreen Version

Developed by: Nintendo Genre: Role-Playing

LSA: [Pokemon FireRed Version, Pokemon Ruby Version, Icewind Dale: Heart of Winter, F.E.A.R. Extraction Point, Guitar Hero Encore: Rocks the 80s]

LDA: [Pokemon FireRed Version, PQ2: Practical Intelligence Quotient, Kingdom Hearts II, CSI: Miami, Black & White: Creature Isle]

Both models give us games that are similar to the original game by giving us results belonging to similar genres. But LSA results are better here because 3 of 5 of its results are role playing games out of which 2 are from the same series. LDA is able to identify 1 game from the same series.

Game 2: Grand Theft Auto: San Andreas

Developed by: Rockstar Games Genre: Modern Action Adventure

LSA: [Grand Theft Auto Double Pack, Bully: Scholarship Edition, Cars, The Godfather: The Don's Edition, Tom Clancy's Splinter Cell Double Agent]

LDA: [Hitman: Blood Money, Thief: Deadly Shadows, NCAA Football 07, Sam & Max Episode 103, Grand Theft Auto: Liberty City Stories]

LSA results are really impressive in this case. GTA Double Pack (package of GTA 3 and Vice City) and Bully are both developed by Rockstar and are very similar games to San Andreas. Other results also have some matching elements such as weapons or cars. LDA also does a good job to stay in the genre except for NCAA Football 07.

Game 3: Call of Duty

Developed by: Activision Genre: Historic First Person Shooter

LSA: [The Sum of All Fears, Killzone: Liberation, Devastation, Medal of Honor Heroes, Vietcong]

LDA: [25 to Life, Tom Clancy's Politika, Crysis, Return to Castle Wolfenstein, Men of Valor]

Both models have almost only returned first person shooters. Results here are good in terms of staying in the genre, but none of the models was able to return any sequels of our input game.

Game 4: FIFA Soccer 2005

Developed by: EA Sports Genre: Soccer Sim

LSA: [FIFA 07 Soccer, FIFA Soccer 06, NBA Live 2003, NBA Live 07, Inside Pitch 2003]

LDA: [FIFA Soccer 06, World Soccer Winning Eleven 9, Top Spin 2, FIFA 07 Soccer, World Soccer Winning Eleven 8 International]

LSA gives us two other FIFA games and all games are from sports genre. LDA gives us 4 other soccer games and 1 tennis game. Results given by LDA are better here.

Game 5: Forza Motorsport 2

Developed by: Microsoft Studios Genre: GT/Street Racing

LSA: [Crash 'N' Burn, MotorStorm, TrackMania United, Forza Motorsport, TrackMania Sunrise]

LDA: [Forza Motorsport 2, Forza Motorsport, Gran Turismo 5 Prologue, Test Drive Le Mans, Baja: Edge of Control, MotorStorm]

For our last game, both models have returned great results with all games from racing genres, though it should be noted that LDA has returned 2 games from the same series compared to 1 from LSA

From the above results, it is quite clear that both models are good at identifying documents that share common topics. LSA has performed better for the first 3 games while LDA was better in the sports genre. A few of the results given by LDA in the first 3 games stray a little bit from the input game.

Conclusion

We have seen how text classification and topic modelling can be used to build models that can make sense out of a set of documents and help us understand their textual sentiments. These simple models are fast to implement and work very well in many textual mining applications.

We discussed how different variants of Naïve Bayes classifiers can help in predicting class assignments. The results given by our NB classifiers are good, but not good enough for real word applications and therefore cannot be used as a stand-alone model. In stead, it should be used in conjunction with other models. One such model could be a combination of Naïve Bayes with an n-gram text classifier [7], that shall loosen a bit of the independence assumption in NB by considering n-gram word probabilities in place of single words. LDA and LSA on the other hand could be used in a recommendation system, by feeding the topics from these models as input to some prediction algorithm.

Future Work

The study in this project helped us analyze textual sentiments present in the dataset with respect to Gamespot Reviews. The next step would be to use the User Reviews as well for analysis. As user reviews are usually shorter and more inconsistent in terms of grammar, it would be interesting to see how the prediction change for the NB classifiers, though I think that it would improve the prediction results by adding some relevant features. Comparison of topic modelling on Gamespot and user reviews will give a good idea about new features introduced by user reviews.

References

1. http://www.nltk.org/api/nltk.stem.html
2. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
3. http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html#sklearn.decomposition.TruncatedSVD.fit_transform
4. http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html
5. http://www.cs.columbia.edu/~blei/papers/BleiNgJordan2003.pdf
6. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
7. http://dl.acm.org/citation.cfm?id=1757820