Recommender Model for Steam Games

CS 97 Final Course Project

Mark Sorial  
American School of Dubai   
 Dubai, UAE  
 msorial44@gmail.com

Coco Gong  
Troy High School  
Fullerton CA, USA  
cocominggong@gmail.com

Sam Sukendro  
 American School of Dubai  
 Dubai, UAE  
samuelsukendro22@gmail.com

Ari Choe  
 Crean Lutheran High School  
 Irvine CA, USA  
0arichoe@gmail.com

ABSTRACT

With the purpose of enhancing both consumer and producer benefits in the Steam game platform, we utilize the concepts we learned in CS97 to construct a recommender system. In order to recommend similar games to a user based on an input game, we implement a model using the K-Nearest Neighbors (KNN) algorithm and another with the concept of cosine similarity. The resultant experiment demonstrates that cosine similarity is the better-performing model out of the two.

1 Introduction of the Overall Goal and Background

Over the three weeks of UCLA CSSI’s CS97 class, we learned the basics of data science through the implementation of various models on real-world statistics and datasets. A variety of concepts such as prediction, classification, and clustering were demonstrated in vivid examples inside Google Colab notebooks. In order to demonstrate our overall skills developed in this course, we are choosing the topic of data clustering as the central focus of our final project.

Specifically, cluster analysis is the grouping of numerous data points into clusters, with the goal of maximizing intra-class similarity while minimizing inter-class similarity. Our project utilizes this principle in a recommender system for Steam games, such that games with similar tags will be recommended according to user preferences.

In this report, we review our two different models for our recommender system. The first model implements the K-Nearest Neighbors (KNN) algorithm, which calculates the Euclidean distances between data points and makes predictions based on the neighboring data points that are the shortest distances away. Our second model uses the concept of cosine similarity, in which similarity is measured by the cosine of the angle between two vectors. This model does not rely on Euclidean distances, which allows it to be an effective complement to our first model.

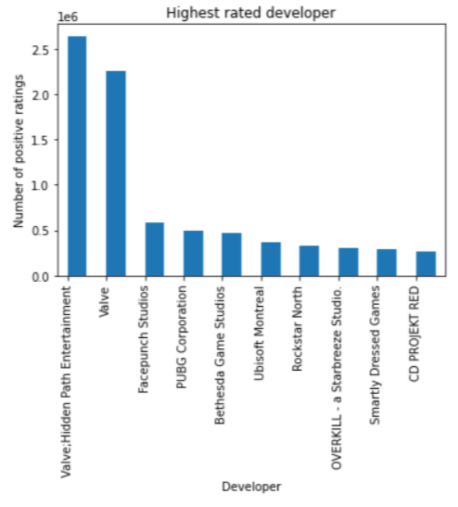
2 Problem Definition and Formalization

With 67 million active monthly users, Steam is one of the biggest online distribution services for video games. It was created by Valve with the initial purpose of establishing automatic updates for their games, but its most prominent feature is the users’ ability to store a myriad of games with access across multiple computers.

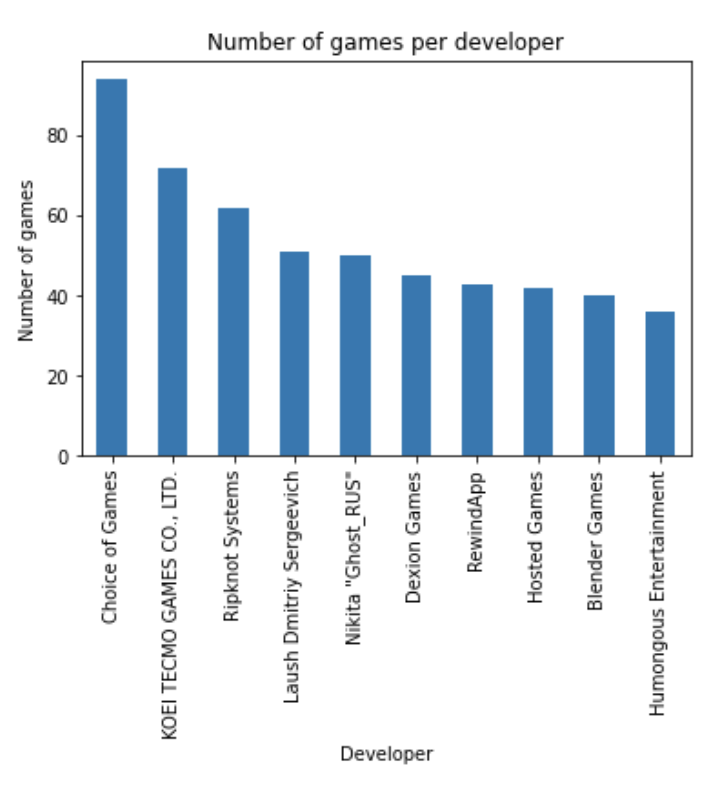
To further enhance the consumer experience of Steam, we decided to implement a model that recommends new games for a user based on a game they played and liked in the past. Simultaneously, this model also aims to assist developers of Steam games by promoting their content in our model’s recommendations.

A recommender system is an algorithm that utilizes user content preference data to make suggestions of other products. Whether the model uses evaluation metrics, e.g., ratings, or implicit derivations from amounts of content interactions, the algorithm always takes in vast amounts of data to fine tune its results.

Before processing our data, we created graphical demonstrations of some interesting statistics we found in the data, as shown below in Figure 1 and Figure 2.



**Figure 1: Highest Rated Developers**



**Figure 2: Number of Games per Developer**

3 Data Preparation Description and Preprocessing

In the process of preparation, we loaded our Kaggle dataset of Steam games into a Pandas DataFrame and analyzed the summary of its data types. We removed certain columns before processing the data in order to eliminate irrelevant data and to adjust our results. For example, we dropped columns that were unrelated to the game content, such as ‘required\_age’, ‘achievements’, ‘owners’, and ‘price’. Additionally, to prevent recommending games that are created by the same developer as the input game, we dropped ‘publisher’ and ‘developer’. Also, all non-English games were dropped to ensure relevance to the audience, which subsequently led to the ‘english’ column itself being dropped. Lastly, we filtered the dataframe with the ratings column to only include games that have received over 500 positive ratings.

Next in our preparation, we combined similar data columns. The Kaggle dataset contained several columns related to the specific content of each game, including ‘categories’, ‘genres’, and ‘steamspy\_tags’. Since there were over 300 tags, we combined these columns for conciseness.

Finally, we set up our organized data for processing by checking for null values and quantifying categorical data. Fortunately, the dataset did not have any null values. However, an issue arose in the processing of categorical data. Since some columns were combined into one, using one-hot encoding resulted in an encoded version of every single combination. Our solution was to use the ‘MultiLabelBinarizer’ from ‘sklearn.preprocessing’ to binarize our labels, so that only the intended combinations were encoded.

4 Methods Description (Detailed Steps)

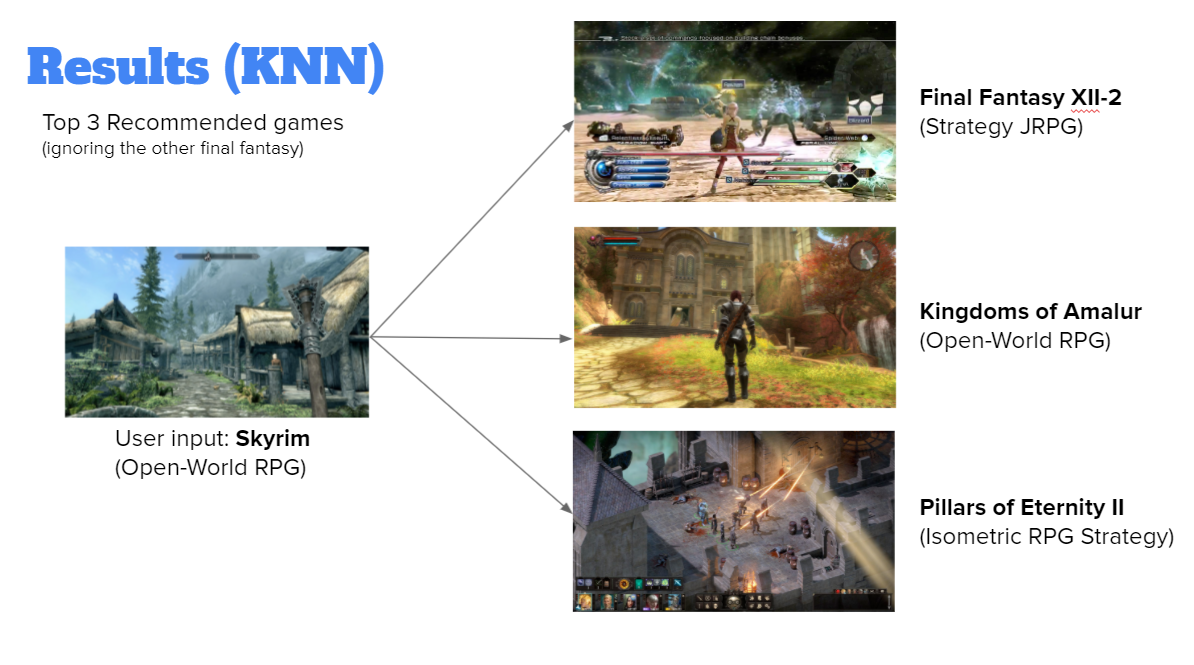
After processing the data, we built the first model with the KNN algorithm. KNN functions by calculating the Euclidean distance between the user’s input game and its “neighboring” games. The Euclidean distances are inversely proportional to the similarities between games; the closer the distance, the more similar the games. The general model we used was ‘NearestNeighbors’ from ‘sklearn.neighbors’. We also used specific terms: ‘n\_neighbors=6’ and ‘algorithm='ball\_tree'’. The number of neighbors specified relates to the number of recommendations returned by our model (the model returns n\_neighbors - 1 = 5 recommendations). The ball tree algorithm is a binary tree data partitioning structure. In terms of clustering analysis, the clusters of data, called balls, are hyperspheres created in a multi-dimensional space. This method’s intended purpose is to speed up the processing time taken by the brute force method.

Our second model uses a different fundamental concept than our first. Cosine similarity measures the cosine of the angle between two vectors in an inner product space. Here, the smaller the angles, the more similar the games. In order to apply cosine similarity, we processed the data again by combining the genres and the tags of our dataset, removing duplicates, converting to lowercase, and concatenating into one column. Then, we converted the resultant collection of text documents to a matrix of token counts with ‘CountVectorizer’ from ‘sklearn.feature\_extraction.text’. The final cosine similarity is applied with ‘cosine\_similarity’ from ‘sklearn.metrics.pairwise’. This model is an improvement from our first, since cosine similarity is less affected by large amounts of dimensions than Euclidean distance. The 300 tags for our games represent 300 dimensions, so as a result, we believe that cosine similarity performed better.

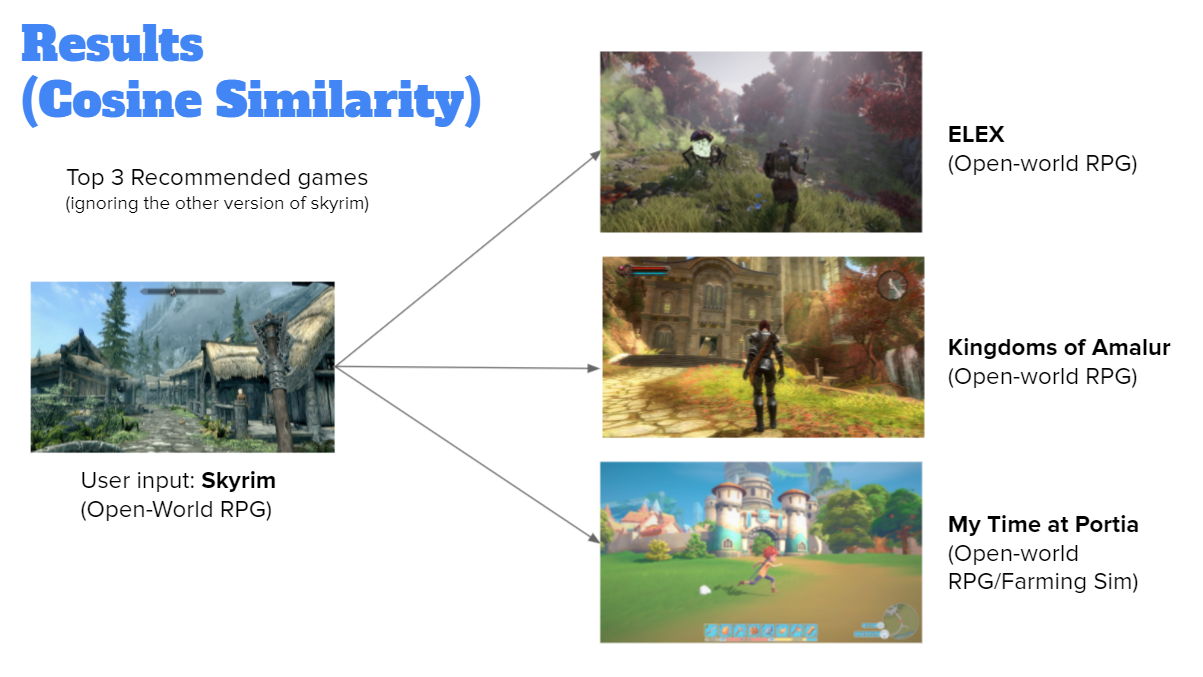
5 Experiments Design and Evaluation

While most predictive models are able to be evaluated with metrics like mean squared error, recommender systems have no ground truth that the results could be compared with. Ultimately, the quality of a recommender system’s results is based entirely on the judgement of the user.

Below is a visualization of our results, as shown in Figure 3 and Figure 4.



**Figure 3: KNN Model Results**

****

**Figure 4: Cosine Similarity Model Results**

However, we did use manual evaluation to help us judge our recommender model. Our evaluation contained two main parts: model performances and algorithmic concepts. To examine model performances, we printed out the tags of the input game as well as the tags of the recommended games for both of our algorithms. Matching tags found in the comparison demonstrate that the model was able to efficiently identify similar games based on content. Our results demonstrated that the cosine similarity model recommended games that had more matching tags to the input. In our second evaluation, we researched online about the details of each algorithm. Our findings proved once again that the cosine similarity model performs better. In data science, a phenomenon called the “curse of dimensionality” refers to the analysis and organization of datasets with very many features and a high-dimensional space. In our research, we discovered that cosine similarity algorithms perform better than KNN algorithms when faced with the issue of the curse of dimensionality. Since our dataset has over 300 game tags (which represents a total of over 300 dimensions), cosine similarity is again a better algorithm for our task.

Although both evaluation metrics rely on manual evaluation, cosine similarity scores are more effective. While the matching tags of our first model are based on user opinion, these scores are quantitative and can be more easily measured. As a result, we decided that the cosine similarity model is the better-performing one.

6 Conclusion

In the evaluation of the performances of our two models, we found that cosine similarity was able to recommend games with more similar tags to the original input. Additionally, in terms of algorithmic concepts, cosine similarity holds an advantage over KNN’s Euclidean distance system since it is able to handle larger amounts of dimensions. Both of our evaluation methods demonstrated that cosine similarity triumphs over KNN in this dataset.

Furthermore, since the results of the cosine similarity system is more easily measured and evaluated due to its quantitative similarity scores, it is also more effective in the aspect of evaluation and comparison. After considering our unique evaluation metric for our models, we concluded that the cosine similarity model is the better implementation of a recommender system out of our models.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2021 Copyright held by the owner/author(s).

ACKNOWLEDGMENTS

We would like to give our sincerest thanks to Professor Yizhou Sun for teaching us about the wonderful realm of data science at UCLA’s Computer Science Summer Institute Program. Through these classes, we learned about various models and their Python implementations in real-world statistics, but we also opened our eyes to the new perspectives in the field of data science. We would also like to thank our Teaching Assistant, Nathakin Lepreecha. In our daily discussions, we reviewed the concepts taught in class with fascinating and humorous methods, and each day was a new step forward in our learning process. Last but not least, thank you to the guest speakers that told us of their own stories in the world of data science. Our experience was greatly enhanced by the practical applications of all of our class materials. With the utmost respect and sincerity, thank you.

REFERENCES

[1] Davis, Nik. “Steam Store Games (Clean Dataset).” Kaggle, 12 June 2019, www.kaggle.com/nikdavis/steam-store-games.

[2] Harrison, Onel. “Machine Learning Basics with the K-Nearest Neighbors Algorithm.” Medium, 14 July 2019, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[3] Prabhakaran, Selva. “Cosine Similarity - Understanding the Math and How It Works? (With Python).” Machine Learning Plus, 22 Oct. 2018, https://www.machinelearningplus.com/nlp/cosine-similarity/.

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00