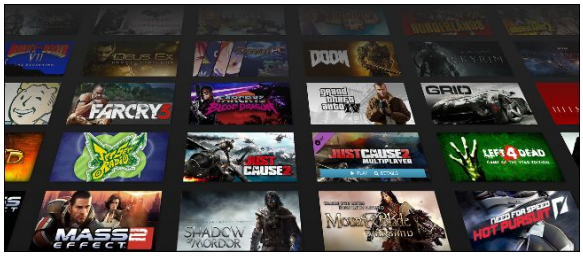
CSE 158: Assignment 2

Predicting a Steam title's price based on a review



Daniil Kubatko

Nayoung Park

Part 1: Dataset identification

Abstract

We present a predictive model utilizing features from a game review dataset in order to explore sales and determine a sellable price tag for a hypothetical game that is to be released. The model is built with Python 3 and is based on natural language processing using TF-IDF(term frequency-inverse document frequency), a statistic that analyzes the importance of a word or a 'term' in the given document.

Introduction

Our group decided to perform the prediction on the *Steam Video Game and Bundle Data*[1], which contains more than 32,000 rows of games registered on Steam, a popular game-distribution platform, along with 50,000 reviews for those games. The training and validation datasets will be built using pairs of games and the correlating reviews to each game in the dataset.

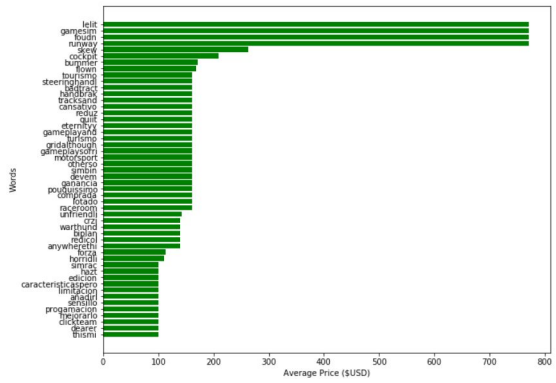
Dataset choice

Due to limited computational resources, we decided to use a smaller dataset with reviews which made us choose a simpler model (see more in the model section). Better results could be achieved if a larger dataset was used for reviews since it would possibly have more overlap with the game metadata dataset.

Exploratory analysis of the data

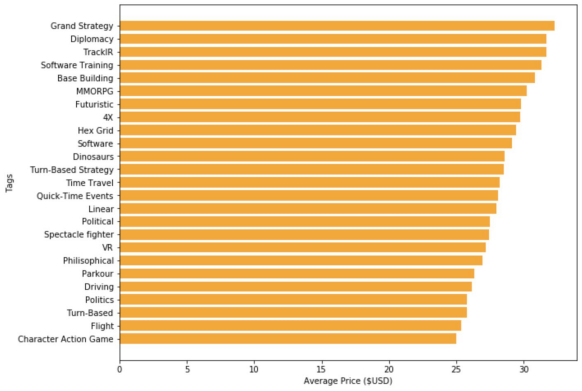
There was a lot of the features that we performed analysis on in the dataset. Below we included the metrics that pivoted the direction of our research the most.

Words to avg. price relation



While there are some obvious relations i.e. the flight simulator reviews tend to correspond to substantially more expensive games, there are also relations that indicate that people tend to be more upset with expensive games since they think they are “disappointing”.

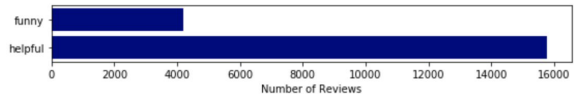
Tags to avg. price relation



This metric would indicate that some types of games are more expensive in general, for example 3D games are more expensive than 2D games, etc.

There are some professional products like game engines that sell at a very high price point, but otherwise our assumptions were correct: the relationship between the tag and the product’s price is clear.

Non-predictive features



Features of the reviews like “helpful” and “funny” showed to be unrelated to the analysis we were trying to perform so we decided to get rid of them later in modeling. Here are the distributions of the features mentioned:

Part 2: Predictive task

Given a (review, game) pair we would predict the price of the game.

The idea behind such prediction would be to indicate a reasonable price for a newly released game. For example, given that the game was only shown to a small group of beta-tester, written review along with the game’s metadata can be analyzed in order to predict a reasonable price point.

A more plausible predictive task would be (ALL reviews, game) pair, however, due to limitations on the size of the dataset (there are only 3k unique titles available), we decided to simplify the task to the one above.

The validity of the predictions will be assessed by both a) MSE and R^2 metrics and b) exploring the weights associated with trends i.e. 3D, shooter, co-op games tend to be more expensive; reviews with words “value” or “overpriced” are clear indications of the possible game price, etc.

Data pre-processing and relation to exploratory analysis

We had to drop almost 20,000 entries of the dataset due to missing features/missing game metadata/unidentified price.

```
f'Missing game info: {no_record}; Missing attribute: {no_attr}; Missing price: {no_price}'
```

```
'Missing game info: 5317; Missing attribute: 4290; Missing price: 10876'
```

In addition, tags along with the review date posted had to be one-hot encoded and reviews had to be mapped to their respective games (in order to construct the (review, game) dataset).

As we saw in the exploratory analysis section, features like “helpful”, “funny”, etc do not have much predictive power due to a low number of reviews that have those features at all.

Instead, we used the tags and sentiment review analysis to make predictions as those clearly indicate certain relations to the price of the title.

Part 3: Model

Given a (game, review) pair, the features to be extracted are the following:

$$\theta * [[\ tf\text{-}idf(review)\ ][\ one\text{-}hot(game\ tags)\ ][\ one\text{-}hot(date)\ ]] = price$$

The model above would be used in a regression model.

The following tools for the class could possibly be considered:

- Least Squares Regression
- Linear Regression (above with optimizations)
- PCA
- Bag-of-words
- TF-IDF

From the options above, we chose the most efficient and relevant tools i.e. LinearRegression, PCA, and TF-IDF. Other tools are just less efficient versions of the ones we chose.

Evaluation and feature choice:

We will then evaluate the performance of the model using MSE and R^2 metrics on the validation set (divided 90%/10% for training/validation).

As for the features extracted from the game metadata/review:

1) TF-IDF is exhaustive for the review analysis
2) From the review metadata, we could use helpful/funny or recommend flags
3) Game metadata doesn't contain many useful features except genres, tags, and early_access

After trying out all the features above, it turns out that funny/helpful are very sparse and don't usually indicate any helpful information about the review.

The 'recommend' tag doesn't affect the performance of the model in a positive way.

Tags overlap with genres significantly however the tags contain more information about the product so we kept those.

Other parts of the game metadata are extremely sparse and are only defined for a tiny subset of the dataset.

The baseline model:

The baseline we chose to compare to would be the same model without TF-IDF analysis, i.e.:

$$\theta * [[\ one\text{-}hot(game\ tags)\ ][\ one\text{-}hot(date)\ ]] = price$$

That would ignore the review analysis at all and will only rely on the game itself.

We also only use the top 100 used tags for the one-hot tags encoding. The reason we do that is due to the majority of the tags (> 350 different tag types) being exclusive for one game so that they don't hold any meaning for other games.

The feature vectors are 118 in size (100 for tags, 6 for the review year 2010 - 2015, 12 for review month) and contain 19 positive marks on average.

The baseline performance:

With normalized linear regression here are the metrics we measured:

$$MSE = 63.44$$

$$R^2 = 0.55$$

Sklearn.metrics evaluation:

```
metrics.mean_squared_error(Y_valid, predictions)
```

```
63.445521437631236
```

```
metrics.r2_score(Y_valid, predictions)
```

```
0.5527165942288355
```

This is quite a high benchmark for the full model we are trying to implement.

The model implementation pipeline:

The pipeline for the feature construction we chose is the following:

Stemming + stopword of reviews -> TF-IDF(reviews, dictionary size = 7000) -> PCA(components # = 100) -> one-hot(top 150 tags) -> one-hot(date posted)

The model performance:

Sklearn.metrics evaluation:

```
metrics.mean_squared_error(Y_valid, predictions)
```

```
55.71212243789392
```

```
metrics.r2_score(Y_valid, predictions)
```

```
0.6072361405169078
```

$$MSE = 55.7$$

$$R^2 = 0.6$$

As we can see, the model outperformed the baseline significantly.

Optimizations/problems:

The biggest problem we faced was overfitting given too many features without PCA TF-IDF decomposition. We also used all of the tags initially which led to issues with overfitting as well. The solution was to only consider meaningful tags (i.e. top N frequent).

There was also a lot of tweaking of the dictionary size for the TF-IDF and # of components for the PCA decomposition. We decided to use an equal amount of features extracted from game metadata and review data. That also led to the best performance of the model.

There was also a problem including free games in the training/validation sets that led to negative predictions by the model. We therefore removed all free games from consideration (around 10,000 reviews) which shrank the dataset significantly. There were also around 7000 reviews without game info or one of the features missing.

Important note:

A better model could be constructed if instead of analyzing a (review, game) pair we would analyze (reviews, game) pairs, i.e. ALL of the reviews for a given game. That would remove a lot of redundancy in the similar feature vectors in the linear regression part.

However, due to an extremely limited amount of unique titles (there are only 3600 games that have at least one review in the dataset) we had to stick with the model described above.

Part 4: Literature

We used the dataset hosted by Julian McAuley named *Steam Video Game and Bundle Data*[1]. It offers 4 different datasets for reviews, bundles and game metadata. We merged one of the reviews datasets and a game metadata dataset. Previous usages of this dataset include *Self-attentive sequential recommendation*[2], *Item recommendation on monotonic behavior chains*[3], *Generating and personalizing bundle recommendations on Steam*[4].

Majority of previous work on this dataset (and in all three of the papers cited) is studying the recommendation system type of predictions. As far as we tried to find, there were no attempts to build a price estimation model based on the same source of data. In particular, there we didn't indicate any attempts to use the review information in order to predict the product's price in the realm of Steam purchases.

There are multiple datasets that also analyze steam purchases, for example *Condensing Steam: Distilling the Diversity of Gamer Behavior* by O'Neill [5].

It analyzes the gamer behavior and does exploratory work without any predictive tasks.

One other example is *Using Steam data to tell you if your game will sink or swim* [6] where the predictive task is to tell whether the game is going to be successful where the results indicated that the number of games published matters the most.

State-of-the-art tools

Price prediction is a very common predictive task in multiple areas (for example, stocks).

The tool that used the most recently is deep learning and, unfortunately, is not taught in this class. Deep learning/neural networks use a different approach to feature extraction so they didn't yield much useful information on what features should be analyzed in particular.

However, more classic prediction tools, specifically linear regression that is usually implemented with least-squares algorithm, has been substantially improved and optimized which we took advantage of.

For example, sklearn's library for python has a lot of optimization and built-in tools such as normalization.

Xgboost is also a very relevant tool for gradient boosting, we didn't end up using it, however.

As for the text analysis and text feature

extraction, NLP has multiple up-to-date tools that still use bag-of-words or TF-IDF with some kind of optimizations under the hood. They also incorporate bi-grams/n-grams approaches and such which we thought would be an overkill for the feature construction for our task.

Part 5: Results and conclusions

The best performance was achieved by using all of the tools considered: TF-IDF, PCA, and linear regression. The performance was significantly worse when using only a subset of those tools.

There was also no improvement when increasing a dictionary size beyond 7000 words or using more than 150 tags.

Observations:

Temporal information about when the review was written indicates that the price of the game is higher more recent review is (which was expected);

Games with reviews in Spring/Summer tend to be more expensive as well.

These are the tags that tend to influence the price a lot:

```
tag_weights[:5]
```

```
[('Episodic', 33.623113175726445),
 ('Violent', 18.464525365245247),
 ('Memes', 9.099876144651885),
 ('Top-Down', 9.06766929456968),
 ('Perma Death', 8.446572632562361)]
```

However, we should keep in mind that those tags are very niche and only appear in a couple of games. Therefore here are the weighted statistic on the tags:

```
r[:5]
```

```
[('Online Co-Op', 48157.02423792303),
 ('Moddable', 42320.20943002208),
 ('Adventure', 39078.63796411132),
 ('Third Person', 34113.19623629146),
 ('Tactical', 32968.240913991474)]
```

As for the review content, after the PCA decomposition it's hard to indicate which words tend to have the most influence on the price.

Conclusion:

Even though the model does perform quite well, there is a major flaw in the logic of predictions: the (review, game) pairs are quite often for the same game and therefore reinforce game-related features to have more weight in prediction (since the price stays the same when reviews are for the same game).

As was described in the “Important Note” part, the best version of this model would be ALL of the reviews for the game packed with the game metadata, however, there was not enough data to perform that kind of study. The model we came up with, however, is totally suitable for both and would certainly perform well-given data in the plausible format.

Therefore, we are quite happy with the work we’ve done and learned a lot about data formatting/preprocessing/analysis and decomposition.

References

[1] Julian McAuley, 2017. Steam Video Game and Bundle Data

[2] Wang-Cheng Kang, Julian McAuley, 2018. Self-attentive sequential recommendation. *ICDM*, 2018.

[3] Mengting Wan, Julian McAuley, 2018. Item recommendation on monotonic behavior chains. *RecSys*, 2018.

[4] Apurva Pathak, Kshitiz Gupta, Julian McAuley, 2017. Generating and personalizing bundle recommendations on Steam. *SIGIR*, 2017.

[5] Mark O'Neill, Elham Vaziripour, Justin Wu, and Daniel Zappala. 2016. Condensing Steam: Distilling the Diversity of Gamer Behavior. In Proceedings of the 2016 Internet Measurement Conference (IMC '16). ACM, New York, NY, USA, 81-95.

[6] Michal Trněný. Using Steam Data to Tell You If Your Game Will Sink or Swim. *VentureBeat*, VentureBeat, 29 June 2017, venturebeat.com/2017/06/28/using-steam-data-to-tell-you-if-your-game-will-sink-or-swim/.