HELPING GAMES REACH OUT
- Predicting the Helpfulness of Comments for Games on the Steam platform

By: Shutong Li and himself :(((((((((

Introduction:

Steam is a video game platform that offers myriad types of video games ranging from 3A games from multi-billion dollar companies to indie games from studio of a handful people. When it comes to purchase decisions, the review sections of the Steam store page for every game offers a very informative and effective way for a potential buyer to overview other's opinion on the game and the game's overall rating. A majority of the review section consists of reviews that are considered "helpful" defined as reviews that stir interaction among users, i.e. having many people rating the review funny/helpful/unhelpful or having people commenting under the review (below screen shot is an example of the review section layout taken from the store page of Sid Meier's Civilization VI).

Currently, when there are no reviews sufficiently rated by users, steam only shows the reviews in chronological order. The goal of this project is to identify potentially helpful comments to put in the helpful section before there are enough reviews sufficiently rated. By pre-selecting helpful reviews before they are humanly voted out, it shortens the cold-start period of the review section, giving potential buyers helpful reviews sooner and thus potentially increase the potential sales of games.

Dataset cleaning and EDA:

There are mainly two dataset employed - one containing reviews made by every steam user collected (called review dataset from now on) and the other one containing information of the games owned by each user (called user dataset from now on).

1) The review datasets comes in compacted json forms as shown below

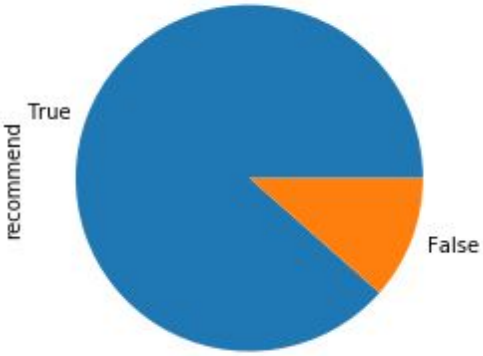
	reviews	user_id	user_url
0	[{'funny': '', 'posted': 'Posted November 5, 2...	76561197970982479	http://steamcommunity.com/profiles/76561197970...
1	[{'funny': '', 'posted': 'Posted June 24, 2014...	js41637	http://steamcommunity.com/id/js41637
2	[{'funny': '', 'posted': 'Posted February 3.',...	evcentric	http://steamcommunity.com/id/evcentric
3	[{'funny': '', 'posted': 'Posted October 14, 2...	doctr	http://steamcommunity.com/id/doctr
4	[{'funny': '3 people found this review funny',...	maplemage	http://steamcommunity.com/id/maplemage

. It is then expanded out into a dataset more suited for a tabular format having each entry corresponds to one review made by one single user.

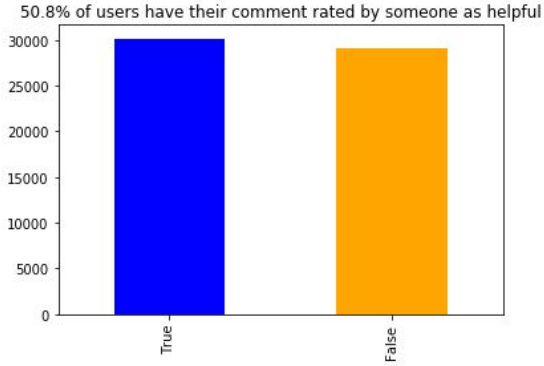
	funny	helpful	item_id	last_edited	posted	recommend	review	user_id
0	NaN	No ratings yet	1250	NaN	2011-11-05	True	Simple yet with great replayability. In my opi...	76561197970982479
1	NaN	No ratings yet	22200	NaN	2011-07-15	True	It's unique and worth a playthrough.	76561197970982479
2	NaN	No ratings yet	43110	NaN	2011-04-21	True	Great atmosphere. The gunplay can be a bit chu...	76561197970982479
3	NaN	15 of 20 people (75%) found this review helpful	251610	NaN	2014-06-24	True	I know what you think when you see this title ...	js41637
4	NaN	0 of 1 people (0%) found this review helpful	227300	NaN	2013-09-08	True	For a simple (it's actually not all that simpl...	js41637

From here we can extract some rudimentary insight about steam reviews. The first thing I

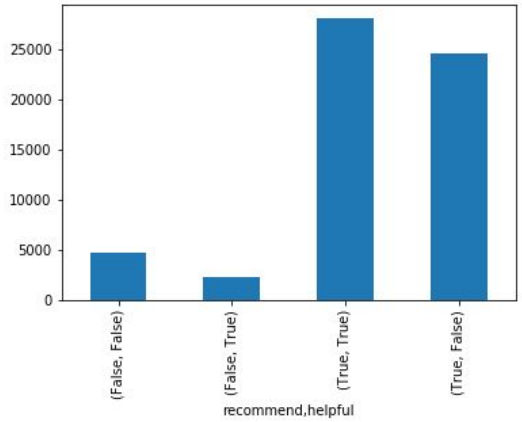
found out is that most reviews are positive.



The distribution implies that whenever a user bothers to write a review, the reason is usually that the user wants to write something nice about it. The next thing to study is the proportion of reviews that are interacted with at all by other users.



The result obtained suggested that almost half of the reviews have never been interacted once. The logical next step of exploration is to check whether or not the proportion of the interacted comment depends on whether the review is a thumbs up or thumbs down.



. The result indicates that when a review is positive(gives thumbs up), the proportion of interacted comments is higher than that of the uninteracted ones, while it is the opposite for the negative reviews. A hypothesis test (more precisely a permutation test) is needed here to statistically prove the significance in difference but I'm not actually writing a paper so I didn't bother, the difference in the distribution shape is very obvious. (very likely a p==0.0 result if test performed).

2) Moving on to the user dataset. Most insights are gained through analyzing why certain data is missing and leveraging the data collection mechanism. The first insight gained is that there are duplicate entries about the same user with the exact same information in the dataset.

	items	items_count	steam_id	user_id	user_url
94	[{'item_id': '320', 'item_name': 'Half-Life 2:...	51	76561198104187800	Rivtex	http://steamcommunity.com/id/Rivtex
5674	[{'item_id': '320', 'item_name': 'Half-Life 2:...	51	76561198104187800	Rivtex	http://steamcommunity.com/id/Rivtex

The duplication might happen due to the fact that user profile is scraped from a game to game basis and duplicate appearance is bound to happen. Another interesting finding is that many users, even though they clearly have made multiple reviews of different games, does not appear to own any game at all. According to Steam policy, one can only review games that he/she

owns. What caused this inconsistency?

funny	helpful	item_id	last_edited	posted	recommend	review	user_id
NaN	0 of 1 people (0%) found this review helpful	251570	NaN	2014-08-19	True	This Game is awesome, yeah theres bugs but it'...	76561198024978857
NaN	No ratings yet	417860	NaN	2015-12-23	True	I cried myself to sleep after the ending	76561198024978857

(above picture is an example of user that has 0 items according to user dataset but appears to have made 2 reviews in the review dataset).
The explanation to the inconsistency turns out to be the limitation of the scraping mechanism and the influence of Steam's privacy policy - users that set their profile to private will not have the information of their owned game scraped, therefore appearing to own no games in the user dataset. The cleaning decision here is to remove the users with private profile. This decision is important to note since it may propagate bias by removing certain groups of people with a specific behavior.

Predictive Goal:

1) Definition of label to predict: As stated in the introduction and in the EDA, it would be helpful to know whether a game review can potentially stir interaction because said ability is considered a trait that indicates helpfulness of the review. For our predictive task I define helpfulness as a binary label of true and false. The label of a review is considered true when the review has at least one user rated it as helpful/funny.
2) Assessing validity: Since the label comes in a very decent 50/50 division, we know that a naive model that uniformly chooses between true and false will achieve an accuracy around 0.5. This will serve as a sanity check to see whether or not the model is functional by checking whether or not the accuracy is statistically more significant than chance.
3) Features to use: As mentioned in the EDA section, the first feature that would come into play is whether or not the review gives a thumbs up or thumbs down to the game. As seen in the previous analysis, a thumbs up is more likely to stir interaction, meaning that this feature shall be considered as a predictor. Another feature to be considered as a predictor is the actual content of the review. It is plausible that what a user says in the review wins him more votes, whether it's funny or (un)helpful. Finally, information about the user should be considered as well, as it is reasonable to assume that it is easier for some users to write reviews that stir interaction than others, whether they have a knack for it or due to other reasons. The information of the user to consider for model training includes playtime within recent 2 weeks, total playtime of the game and the user_id itself (emphasizing the person factor that makes a review helpful).
4) Feature preprocessing: First and foremost the joining of the user dataset and review dataset is required in order to connect user information with review information. In order to join the two datasets, a join key of (user_id, item_id) pair is devised to uniquely identify every user-item-review combination. The resultant joined table looks is shown in the picture below:

posted	recommend	review	user_id	user_item	item_name	playtime_2weeks	playtime_forever	helpful_vote	labels	review_length
2011-11-05	1	simple yet with great replayability in my opin...	76561197970982479	(76561197970982479, 1250)	Killing Floor	0	10006	0	False	45
2011-07-15	1	its unique and worth a playthrough	76561197970982479	(76561197970982479, 22200)	Zeno Clash	0	271	0	False	6
2011-04-21	1	great atmosphere the gunplay can be a bit chun...	76561197970982479	(76561197970982479, 43110)	Metro 2033	0	834	0	False	39
2014-06-24	1	i know what you think when you see this title ...	js41637	(js41637, 251610)	Barbie™ Dreamhouse Party™	0	84	20	True	110
2013-09-08	1	for a simple its actually not all that simple ...	js41637	(js41637, 227300)	Euro Truck Simulator 2	0	551	1	True	116

Moving on to individual feature transformation: For thumbs up/down feature (denoted as "recommended"), the only preprocessing required is to transform the

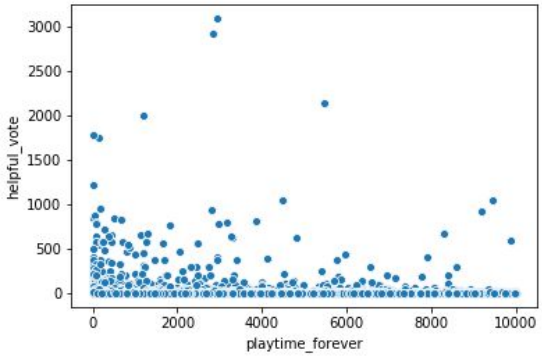
feature into a numerically binary one, 1 for thumbs up and 0 for thumbs down suffices. The feature of user_id requires one-hot encoding. The feature of review text requires some text parsing. The methodology used in this project is to clean out punctuation and upper/lower cases and then vectorize the texts into a TF-IDF matrix. The rest of the features are innately numeric and can be left as is. Below is a screenshot of the column transformers used on the dataset.

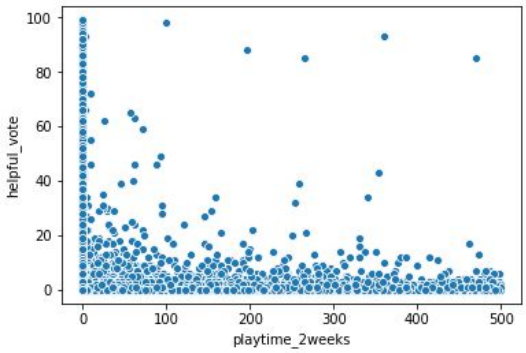
```
transformers = [('parse_review', TfidfVectorizer(), 'review'),
                ('parse_item_name', OneHotEncoder(handle_unknown='ignore'), ['user_id'])]
ct = ColumnTransformer(transformers=transformers, remainder='passthrough')
```

5) Model selection: Since it is a binary classification problem, models that will be used in this project will be Logistic regression and a support vector classifier. The reason to use these two is that one focuses on the distinction of classes “at large” while the other one focuses on maximum margin between the two classes. The expectation for the comparison of the performance between the two models is that either logistic regression captures the overall pattern and SVC overfits, or svm captures the pattern’s nuance while logistic regression is insignificant.

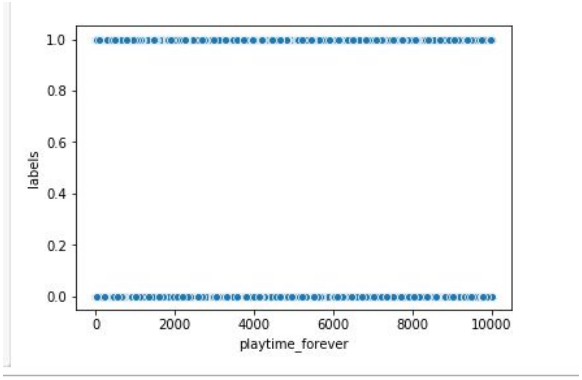
Model Selection:

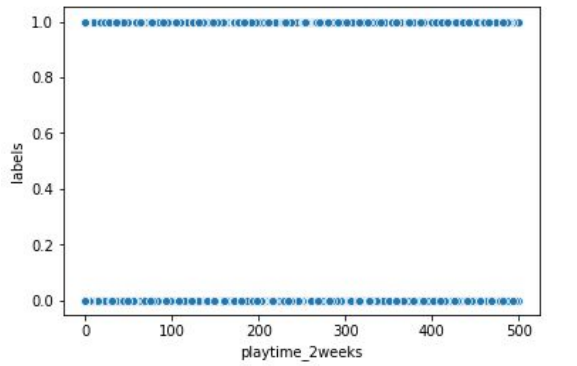
1) Model selection: As stated in the previous section, 2 models will be used in this predictive task to account for different cases of the distribution of data points in the feature space. Logistic Regression will be used to generalize a more separable case and SVC will be used to capture a more nuance difference. In the case of these two models, optimizing the hyperparameters is not as important as constructing meaningful features, but for the sake of optimization, Logistic Regressor can be optimized via sklearn LogisticRegressorCV and SVC will be tuned via GridSearch (or manually).
2) Trials and Failures: after multiple trials, the feature of playtime within 2 weeks and total playtime of a game turns out to be counter effective for the predictive task. Below is a 2-D visualization to show why it is so messy.





Here the binary label is expanded back into the discrete values that it used to be to better show how random a review would be voted regarding the playtime committed by the writer of the review. Below is the distribution of playtime vs the actual label:





It is quite apparent that this is really not an ideal predictor for the label. The reason this distribution is not analyzed prior to the feature engineering is that this analysis can only be done after the actual features are generated. The previous EDA only works with individual or aggregation of existing features. In short, removing the playtime features appeared to improve the result.

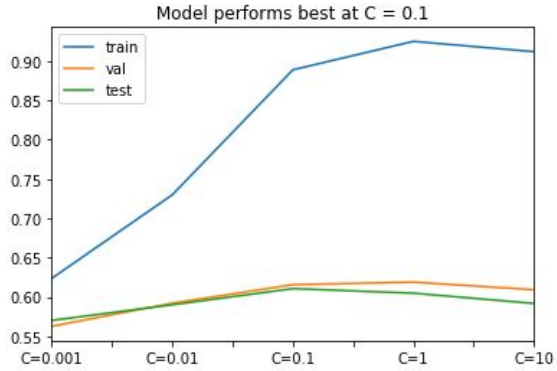
3) Final feature selection: the final features used are ['review', 'user_id', 'review_length', 'recommend']. These features seem to provide the best performance for both models in all cases.

4) Issues during model fitting: There does not seem to be issues regarding either scalability or overfitting, possibly due to the model in use.

5) Performance Comparison and analysis:

	train	validation	test
logistic_regression	0.547530	0.533482	0.544999
LinearSVC	0.887500	0.628289	0.620183
Deg2_SVC	0.524731	0.508144	0.524727

Even though both logistic_regression and LinearSVC is statistically significantly better than a random naive classifier, it is apparent that LinearSVC performs significantly better than the Logistic Regressor. This behavior can be explained by the previous assumption that the difference between the data points with the two labels are more nuance than general. In other words, the data points don't cluster in an apparent linearly separable fashion. Instead, the difference is more subtle and is better captured with a max margin classifier like SVC. And for the Linear SVC specifically, the model performs best when the regularization constant is between 0.1 to 1.



Literature Reference

Aside from the papers given along with the Steam dataset provided on the course webpage, there are almost no paper that I can access without paying extra money. Below are the reference to the paper from which I utilize the same dataset.

Self-attentive sequential recommendation

Wang-Cheng Kang, Julian McAuley
ICDM, 2018

Item recommendation on monotonic behavior chains
Mengting Wan, Julian McAuley
RecSys, 2018

Generating and personalizing bundle recommendations on Steam
Apurva Pathak, Kshitiz Gupta, Julian McAuley
SIGIR, 2017

Liang, Xiaohua, and Siyu Yang. *PC Game Play Time Estimation Based on Steam Data and Reviews*.

1) Description of dataset and its use in the original works: The dataset was originally used in the paper mentioned above for creating a recommender system for bundle/item recommendations.
2) State of the art method for this dataset: For one of the studies, the researchers utilized a new sequential model called *Transformer* instead of some traditional models such as Markov chain to train a sequential recommendation process. In another paper, they amps up the granularity of data and incorporates both explicit and implicit user feedback to create a recommender system based on a monotonic behavior chain. (Yeah I don't know what I'm talking about)
3) Other similar dataset: According to the last citation, there is an attempt to predict steam play time based on game and user information. However, as similar as the features involved to the ones used for the predictive task of this project, the predictive goal makes the two projects very different and none of the feature engineering of that paper helped in this project.
4) Similarity between other works conclusion: as stated above, I have no access to the papers that actually did anything similar to this project, therefore I have no way of knowing how they defined helpfulness or what features they engineered or what conclusion they arrived at.

Conclusion

1) Result and Conclusion: In short, it seems that it is reasonable to distinguish whether or not a comment will be helpful based on the features involved. However, more complex feature engineering may be required to obtain better results at predicting what leads to a helpful comment. More specifically, the TF-IDF vectorization can be switched to a more complicated feature transformation that involves the information of the game and the time of the review to leverage the semantics of the text. An example to justify the suggestion, comments about a First Person Shooter game may require a very different content from a review about an RPG to be considered helpful to the potential buyers. To do this requires not only better algorithms but also data about the games. Currently there are only information about users and reviews, information on the genres/number of sales/publisher of game can help as well in the classification process.
2) Feature representation performance: As discussed in the model selection section, certain feature (playtime of a game from certain user) is not as good a predictor as expected. Thumbs up/down, TF-IDF of the review text, One-hot of user_id all helped
3) Interpretation of Parameter: The linearSVC that performs best requires a

error penalty 0.1< C < 1. This means that we want to be lenient on the wrong data points the model has to give up in the hope of finding the max margin. This behavior also implies that there are too many mixed cases in our current feature space and having the error penalty too high would only work against our current model.
4) Why did the proposed model succeed and why others failed: As stated (countless times) in both model selection section and predictive goal section. SVM and Logistic Regressor are suitable for different cases of the data distribution, and in this particular scenario, the difference between the two community of label is more nuance than general therefore the classification of such label benefits most from models that looks for maximum margin such as a Linear SVC.