Formulation of business objective

The goal of the project was to increase time spent by users on social platform reddit.com Naturally more time spent overall by users on reddit.com would increase ad-revenue of the platform, which is the main source of income for the website. As Netflix showed, one of the best ways to keep users on the website is to create an intelligent recommendation system of content, so that if a given user reads all the newest posts from subreddits that he likes he won't leave the website, but will be recommended more content to digest. That's why we decided to create a subreddit recommendation system based on users' comment history. In the project we will explore 2 vastly different recommendations approaches. Additionally it is worth mentioning that such a system would increase user satisfaction making it much easier for new inexperienced users to assimilate into the platform.
(Results can be also analyzed scientifically by sociologists to find clusters of users interest and using them find interesting patterns.)

Selection of measures

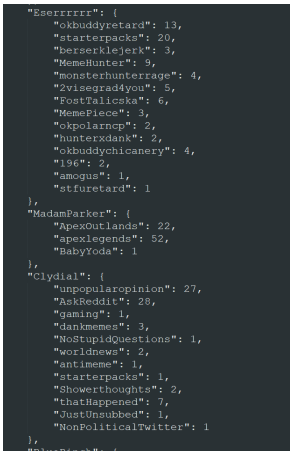
The best evaluation of our system would be to measure user satisfaction with recommendations. However, with our limited time and resources, we decided to select some random set of users and see if recommendations are reasonable. For association rules, we have also used Lift as a measure of the attractiveness of a given rule. For clustering, we have calculated the silhouette score and examined generated clusters to see if subreddits in those clusters can be described by some keyword e.g. rap,film or games.

Data Selection

The project required to gather data on users preference, the only viable way to do that with our access to reddit, was to scrape users comment (or posts) history.

Our first approach was to choose an open-source Reddit datasets provided by a wonderful project called Pushshift. Unfortunately there were a couple of problems with these datasets. Firstly they were a bit outdated, secondly and more importantly, data scrapped for them was done in the comment/post by comment/post manner, and we required data scrapped user by user (We required all the comments made by a given user of a set of users). It was possible to dig out such data from the aforementioned datasets, but it was extremely inefficient (around 1 megabytes of dictionaries describing users for every 100mb of data). It introduced a bias to our dataset, because the dataset was scrapped comment by comment, dictionaries describing given users had missing values. That's why in the end we decided to gather data on our own using Reddit API. Which was neatly implemented in Python via a library called "Praw".

The dataset was constructed in the following way:
It is a dictionary containing rows of dictionaries containing comment information about each user. For example “MadamParker” posted 22 comments on ”ApexOutlands” subreddit, 52 comments on ”ApexLegends” and 1 comment on “BabyYoda” subreddit.



Data preprocessing

Scrapper

Scrapping proved to be difficult due to Reddit api hard-limits. The hardest thing was to generate a stream of fresh users. At first we were going from subreddit to subreddit scrapping post by post and in those posts we were scraping comment by comment. This proved to be inefficient. Later we found out that there is a possibility of opening a live stream of comments using the PRAW library, so we did that by scraping author by author of comments provided by the stream. We scrapped the data from r/all stream, hoping that we avoid any bias created by scraping users only from a certain pool of subreddits. Scrapper was implemented both on one process, and also using Python multiprocessing library, unfortunately, both of the methods were often crushing, slowed down for no reason, and required user input around every one hour of scrapping. But in the end we created a fresh dataset large enough to reach decent results.

Matrix preprocessing

Problem: Transforming 3D dictionaries to tabular data with meaningful information.

The easiest approach to transform 3D dictionaries to tabular data would be to use Pandas which provides the solution right out of the box. Unfortunately, creating sparse matrices from dictionaries in pandas is not optimized. Taking a different approach and writing our own transforming functions we end with the process which is over 200 times faster. After transformation, we obtained a data frame with columns as subreddits, rows as users, and values as a number of comments in a given subreddit.

In the next step, we performed initial filtering of the data in order to reduce the noise:

- Removed rows and columns where all values were not greater than 5.
- Extracted the first 2000 most popular subreddits and removed 3 subreddits with the highest occurrences to remove the skewness of the data while preserving the amount of information.

For the Association rules we did a bit different preprocessing:

- we transformed the table to binary giving 1 if the user wrote more than 2 comments and 0 otherwise

- We removed the 20 most popular subreddits and chose around 5000 or 10000 most popular ones

Data version control

When it comes to DVC we used it exclusively to create a pipeline that'd make the whole process easier to mantain (since it will 'skip' steps that didn't change) and more organized (instead of creating e.g. one file with everything the DVC pipeline makes it easier and safer to split our code into certain parts).
We didn't add all the files that we were working with to the pipeline. For example, we used the scrapper only a few times to create the database and since this process takes a lot of time and we don't need to repeat it often - we decided to leave it out of the pipeline.

Application of DM algorithms

Association rules

Our first approach was to use association rules so that we could have simple rules telling users that if he likes a given set of subreddits he might also like the one recommended by us.

With the first dataset containing data from 1 month, we did not get promising results. The recommendations were poor, probably, because we did not get full information about any of the users. (Only from one month)

After switching to our own dataset results were more promising. However, we still have the problem that lots of recommendations involve some obvious suggestions like /AskReddit which is a subreddit that most users used. Similar subreddits were involved in so many rules, that we were running out of memory before we could get more interesting predictions.

After removing them we were able to get better predictions that made sense. (Like suggesting another game to a user that is commenting on lots of game subreddits). However, most of the predictions were simply giving the user more popular reddits that were related to currently commented by the user. It is due to the fact that till this moment we were choosing the most appropriate rules by linear combinations of support and confidence.

Then we decided to switch to a metric called "lift", which tends to favour less popular item sets, which is exactly what we needed. At the same time, we managed to obtain a bigger dataset with ~30 000 users and more subreddits so we were able to select the top 5000 and top 10000 subreddits for our recommendations. The results were much more promising. We managed to generate 1 000 000 rules that were meaningful in some way.

To somehow fight the problem of limited memory we have also implemented an SQL database for storing generated recommendations. That way in the future if we generate more rules we will be able to store them in a Database and to easily serve recommendations to users via the Web. However, like always, there are no free lunches. Our SQLite

implementation of the database was using less RAM, but for 1 mln rules query time was 2.0s, and when rules were stored in pandas df execution time was ~ 60ms.

We also want to point out that while generating rules we limited the maximum size of frequent itemset to 7 items. The main issue in generating rules was our limited computing power/RAM so all the parameters mentioned above are a compromise between quality and reasonable computing time. We know that we could get more interesting association rules by setting e.g. lower minimal support for frequent itemsets, but then we would need to buy more RAM or decrease the number of chosen subreddits.

Dimension reduction

Dealing with sparse data the first and obvious step is to reduce the number of dimensions in the data to prevent the occurrence of the course of the dimensionality. Struggling with choosing the type of the mapping algorithm we decided to test and pursue the two most promising methods, namely: t-SNE and PCA. We choose t-SNE for its ability to preserve the local neighbourhood, and PCA as it is a purely linear based method with the ability to map new unseen points to the previously created lower-dimension space.

Clustering

DBSCAN

The distribution of the obtained data was a sphere (in case of mapping to 3D), so immediately we decided to give up the idea of using KMeans as we would not be able to discover the true number of clusters and the silhouette score suggested using only one cluster. We ended up using the density-based clustering method, which turned out to give encouraging results. Although the silhouette score was not an ideal metric for our data distribution we still decided to use it but more as a suggestion rather than a formal metric.

KMEANS

Being satisfied with the results from the non-linear mapping we tried PCA. Knowing the approximation of the true number of the clusters we were able to perform KMeans clustering with the number of clusters in the neighbourhood of the value obtained from DBSCAN.

For fine-tuning, we decided to implement the algorithm calculating silhouette score for a given epsilon (eps) and a minimum number of samples (min_samples) or a number of clusters in the case of KMeans. In order to make the algorithm faster, we implemented “skip jumps” which made the iterations much faster by skipping the parameters without the improvement in the score. In the end, we choose the parameters with the highest score and a rational number of the clusters (we fixed this value as 800, but ultimately we will calibrate this value according to the user satisfaction score).

Results

The best way to measure the results would be to set up an internet service where users could rate recommendations based on different systems and compare them. Sadly, we neither had enough time nor resources to do so. However, we are planning to do it in the future.

Association Rules

We decided to show the results of 2 final sets of recommendation rules. The first one contains the top 5000 subreddits, and the 2nd one 10000. We removed the 20 most popular. Here are some example recommendations for users (Each time we selected up to 10 most recommended subreddits):

5000

```
User likes: {'gothsluts', 'GirlsGoneWilderness', 'u_Unknownpleasures89', 'Yololaceandlingerie', 'cougars_and_milfs
_sfw', 'milf', 'bigasses', '18_19', 'GaybrosGoneWild', 'facesitting', 'collegesluts', 'assinthong', 'nj4nj', 'Teen
Beauties', '2000sGirls'}
User should like: ['cock', 'BiCuriousGuysChat', 'LegalTeens', 'DadsAndBoys', 'MassiveCock', 'gaynsfw', 'ThickDic
k', 'ratemycock', 'gayporn', 'CutCocks']
User likes: {'AdorableOnlyfans', 'onlyfansgirls101', 'FreeOnlyFansPage', 'onlyfanschicks', 'HotOnlyfans', 'HornyOn
lyfans'}
User should like: ['OnlyFansPromotions', 'GoneWildOnlyfans', 'Onlyfanspromoss', 'OnlyfansXXX', 'naughtychicks', 'N
aughtyOnlyfans', 'OnlyFansLifestyle', 'SexyOnlyfansGirls', 'OnlyFansBusty', 'promoteonlyfans']
User likes: {'Battlefield6', 'Amd', 'pathofexile', 'battlefield2042', 'PathOfExileBuilds'}
User should like: ['Battlefield', 'intel', 'overclocking', 'buildapcsales', 'pcmasterrace', 'hardware', 'gaming',
'BattlefieldV', 'nvidia', 'hardwareswap']
User likes: {'deeplearning', 'AskComputerScience', 'softwarearchitecture', 'learnprogramming', 'MachineLearning',
'Chodi', 'learnmachinelearning', 'math', 'linuxquestions', 'crypto', 'AskProgramming', 'AskPhysics', 'computerscie
nce', 'compsci', 'algorithms'}
User should like: ['DesiMeta', 'linux4noobs', 'learnmath', 'IndiaSpeaks', 'linux', 'india', 'cscareerquestions',
'ProgrammerHumor']
User likes: {'mildlyinfuriating', 'juggling', 'hermitcraftmemes', 'AskReddit', 'interestingasfuck', 'cursedcomment
s', 'nextfuckinglevel', 'Unexpected', 'memes', 'BrandNewSentence', 'okbuddyretard', 'starwarsmemes', 'oddlyterrify
ing', 'ChoosingBeggars', 'MetalMemes', 'ConservativeMemes', 'trackandfield', 'Meshuggah', 'darkjokes'}
User should like: ['blursedimages', 'youngpeopleyoutube', 'oddlysatisfying', 'Whatcouldgowrong', 'greentext', 'Hol
Up', 'facepalm', 'redditmoment', 'Damnthatsinteresting', 'copypasta']
User likes: {'UFOscience', 'UAP', 'space', 'ufomemes', 'australia', 'UFOs', 'CoronavirusDownunder', 'brexit', 'UFO
believers', 'UF0', 'ufo', 'HighStrangeness'}
User should like: ['aliens', 'ukpolitics', 'nextfuckinglevel', 'mildlyinteresting', 'todayilearned', 'SpaceXLoung
e', 'funny', 'Futurology', 'Showerthoughts', 'pics']
```

10 000

```
User likes: {'gothsluts', 'GirlsGoneWilderness', 'u_Unknownpleasures89', 'Yololaceandlingerie', 'cougars_and_milfs
_sfw', 'milf', 'bigasses', '18_19', 'GaybrosGoneWild', 'facesitting', 'collegesluts', 'assinthong', 'nj4nj', 'Teen
Beauties', '2000sGirls'}
User should like: ['cock', 'BiCuriousGuysChat', 'LegalTeens', 'DadsAndBoys', 'MassiveCock', 'gaynsfw', 'ThickDic
k', 'ratemycock', 'gayporn', 'CutCocks']
User likes: {'AdorableOnlyfans', 'onlyfansgirls101', 'FreeOnlyFansPage', 'onlyfanschicks', 'HotOnlyfans', 'HornyOn
lyfans'}
User should like: ['OnlyFansPromotions', 'GoneWildOnlyfans', 'Onlyfanspromoss', 'OnlyfansXXX', 'naughtychicks', 'N
aughtyOnlyfans', 'OnlyFansLifestyle', 'SexyOnlyfansGirls', 'OnlyFansBusty', 'promoteonlyfans']
User likes: {'Battlefield6', 'Amd', 'pathofexile', 'battlefield2042', 'PathOfExileBuilds'}
User should like: ['Battlefield', 'intel', 'overclocking', 'buildapcsales', 'pcmasterrace', 'hardware', 'gaming',
'BattlefieldV', 'nvidia', 'hardwareswap']
User likes: {'deeplearning', 'AskComputerScience', 'softwarearchitecture', 'learnprogramming', 'MachineLearning',
'Chodi', 'learnmachinelearning', 'math', 'linuxquestions', 'crypto', 'AskProgramming', 'AskPhysics', 'computerscie
nce', 'compsci', 'algorithms'}
User should like: ['DesiMeta', 'linux4noobs', 'learnmath', 'IndiaSpeaks', 'linux', 'india', 'cscareerquestions',
'ProgrammerHumor']
User likes: {'mildlyinfuriating', 'juggling', 'hermitcraftmemes', 'AskReddit', 'interestingasfuck', 'cursedcomment
s', 'nextfuckinglevel', 'Unexpected', 'memes', 'BrandNewSentence', 'okbuddyretard', 'starwarsmemes', 'oddlyterrify
ing', 'ChoosingBeggars', 'MetalMemes', 'ConservativeMemes', 'trackandfield', 'Meshuggah', 'darkjokes'}
User should like: ['blursedimages', 'youngpeopleyoutube', 'oddlysatisfying', 'Whatcouldgowrong', 'greentext', 'Hol
Up', 'facepalm', 'redditmoment', 'Damnthatsinteresting', 'copypasta']
User likes: {'UFOscience', 'UAP', 'space', 'ufomemes', 'australia', 'UFOs', 'CoronavirusDownunder', 'brexit', 'UFO
believers', 'UF0', 'ufo', 'HighStrangeness'}
User should like: ['aliens', 'ukpolitics', 'nextfuckinglevel', 'mildlyinteresting', 'todayilearned', 'SpaceXLoung
e', 'funny', 'Futurology', 'Showerthoughts', 'pics']
```

We know that results are a bit controversial. But there is nothing we can do about the fact that there are a lot of porn-related subreddits among the most popular ones. It is simply Reddit.
Let's discuss some results. For the first user, we can clearly see that he has recommended mostly other porn subreddits. Surprisingly, he got lots of recommendations with gay subreddits, while he mostly commented under those containing women. It is due to the fact that he also commented under 2 gay subreddits. These probably had less support in our dataset so they had a higher lift value. We can clearly see that there is a place for improvement in our metrics. For other users, results are pretty good (except those that had contained subreddits that were not present in our rules). So a person who liked porn, got more porn, a person who liked games and technology got more games and technology. Someone who liked programming and machine learning got recommendations concerning math and IT. What is surprising is that he got 2 India related subreddits, which we suppose is caused by the fact that lots of internet users from India are interested in programming. The last user who liked UFO got only one recommendation concerning UFOs but still, we believe that the algorithm grasped the idea by recommending space and future related sites.

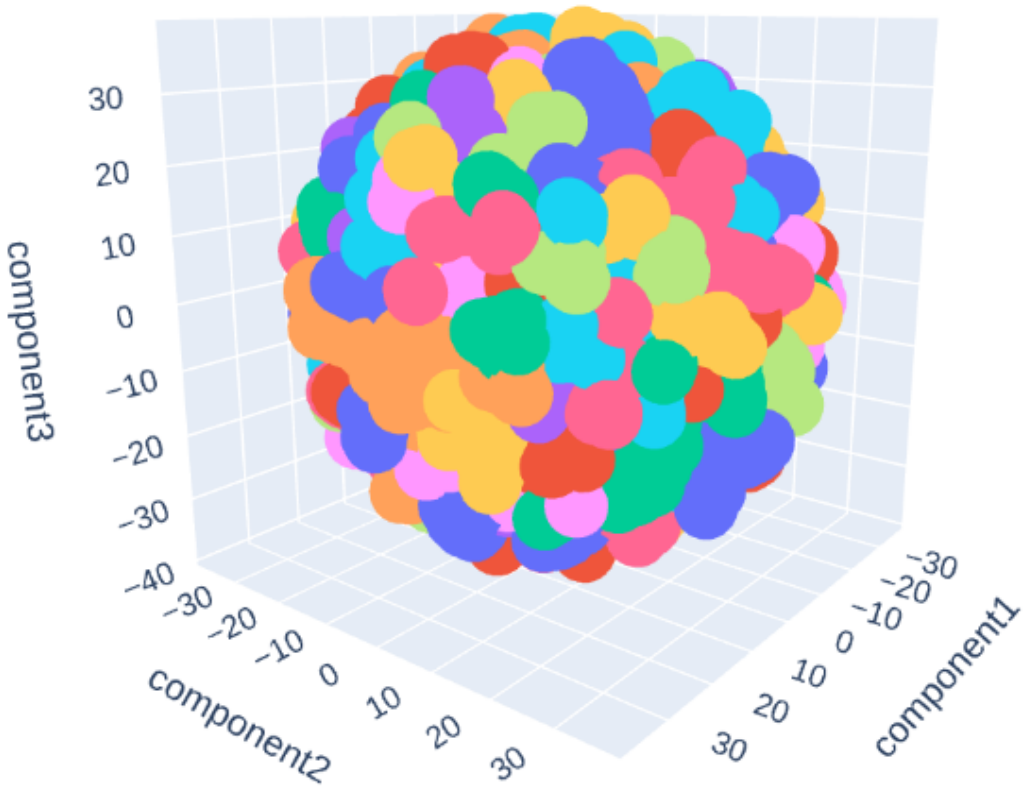
That means that we probably need to gather more UFO-related data. One last issue that we need to address, but still do not know how, is to address the fact that users write more in some subreddits and less in others. Multiplying lifts by the number of comments in a given subreddit is definitely a promising idea of solving this problem that we will try in the future.

CLUSTERING

The idea behind using two different methods of clustering was to compare linear to non-linear approaches and benefit from both of them. We found out that on average using DBSCAN gave better results. Unfortunately because of the random projection performed by t-SNE we are not able to transform unseen points to the projected space. So we came up with the idea of recommending subreddits to new users using clustering based on PCA (KMeans), and then periodically (due to the complexity and time of calculations) inform users

of the cluster they belong to and most popular sites in their clusters. From the 5 most popular subreddits in the clusters, we could easily identify clusters about porn, automotive, movies, politics, memes, cryptocurrencies, stocking and many others. For the whole list of the most popular subreddits, we encourage the reader to visit our github site. In the end, we are quite satisfied with the results, although we still see a space for improvements e.g. larger data or trying approaches like Locality-sensitive hashing.

Visualization



We can see that t-SNE with DBSCAN produced nicely, evenly distributed clusters of users.

Interpretation

We have created a decent porn recommendation system. Being more serious, we think that we have shown that the chosen methods work fairly well on our dataset. However, we know that there is a lot of room for improvement. In the future we will definitely try to gather more data and explore other recommendation systems e.g. try non-user based recommendations. Moreover we certainly need to explore the topic of weighted association rules.

Created by:
Jan Gruszczyński
Mikołaj Kruś
Maciej Filanowicz
Kacper Trębacz