Steam Video Game Recommender System

Gaofeng Peng
Rady School of Management
San Diego, California
gpeng@ucsd.edu

Shihong Hu
Rady School of Management
San Diego, California
s5hu@ucsd.edu

Yuwei Tang
Rady School of Management
San Diego, California
yut001@ucsd.edu

Xingyu Wei
Rady School of Management
San Diego, California
x4wei@ucsd.edu

ABSTRACT

In order to make bigger profit, many companies, especially the internet companies,try to make recommendations accurately and efficiently. By combining item purchase tendency and cosine similarity among users, we develop one recommender system that will successfully predict 87% of the output.

Besides, We also conduct text mining on reviews given by customers about each item adopting techniques of sentiment analysis and make predictions on the helpfulness of reviews as well as potential item scores with the method of linear regression. The results shown in the text mining part suggests strong validity of our models with tf-idf values of common words used to feature the predictors.

KEYWORDS

Machine Learning, Text Mining, Sentiment Analysis, Recommender System, Collaborative Filtering

1 INTRODUCTION

Recommender system is becoming a more and more important tool for improving sales. And one most crucial part in building Recommender system is to understand the relation between users and items. [4]. By working through the purchase data in steam data set, we are able to establish the network between users and items. Looking in to this network, we find some useful clue to build the right recommender system. There exist many ways to build recommender systems, such as *Collaborative filtering*, *Cluster analysis*, *Matrix factorization*, *Deep learning*. Here we don't only rely on one method, but to blend the *cosine similarity between users* with *best sale items list* to make prediction. And it works well.[1] However, if we want to make the predictions for 2,500,000 pair of user-item[2], it will take a long time.

In addition, the characteristics of the provided dataset allow us to implement, apart from basic natural language processing, prediction tasks from the perspective of users' reviews. To predict the potential item ratings, we first select the probability that each item might be recommended by customers as the substitute variable for rating and construct our feature vectors by assigning TF-IDF value of common words to each item. By running the linear regression model, we are enabled to procure predictions on the objective variable.

[1] 87% accuracy, 91% recall rate, 81% precision
[2] number of pairs in Validation set and Test set

2 RECOMMENDER SYSTEMS: DATASET MANIPULATION

We use Steam Video Game and Bundle Data credited by Julian McAuley[3] for training purposes. We merge the 615 bundles data to 88,310 user-item dataset and arrange data to user-item purchase observation. This resulted in a data of 5,153,209 purchases, 88,310 users,12,383 items, and 2,798 genre data in total. See Table 1 for summary of data structure.

Table 1: Steam Video Game Purchase Data

User:	88,310
Total items:	12,383
Total genre:	2,798
Top 20 purchase items:	515,321 (10% of all sales)

2.1 Web Scraping

As we found that only 615 bundles data have genre. For sake of lacking genre data on 9,585 items, we crawl genre data for each item on *Steam* app page using request to get html text by inserting item-id after fix address of 'https://store.steampowered.com/app/' and using bs4 to parse it and collect genre under <div class=' details_block' tag. Moreover, as we find that some of items are video and don't belong to any game genre, we assign its genre 'Not Game'.

2.2 Inspect On Data

Among 12,383 items, we find that top 20 items account for over 10% purchased in dataset. This finding suggests a popular item purchase tendency on Steam, and possibility of personal purchase recommendation system by item's popularity.

Also, among top 20 popular items, 13(65%) of them have Action genre, 5(25%) of them have Indie genre. It indicates that most of popular items have similar genre. This indicates users often buy items that have similar genre to those they owned, and their purchase behavior could be predicted by gen type.

[3] The existing Bibstrip data, copyright text and permission block in the sample file are dummy values, so the user needs to provide the correct values required for the submission in the metadata dialog box.

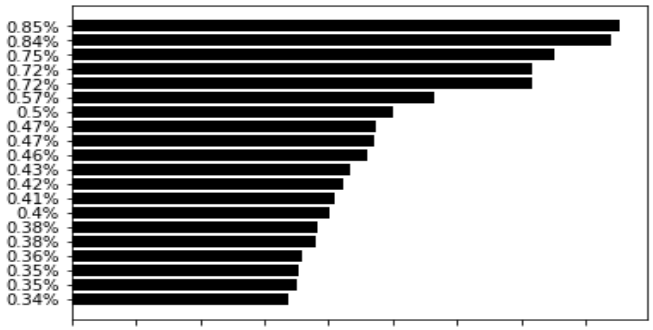

Figure 1: Top 20 best sale games in Austrilia steam store takes 10% of the total sale

2.3 Data Splitting

In order to build model, we need Train set, Validation set, Test set. Train set is for training the similarity, when Validation set decide the threshold value T, which decide the level of similarity that we predict user will buy the item. Test set is for testing the result of model and threshold we previous get.[1]

3 RECOMMENDER SYSTEMS: PURCHASE PREDICTION

3.1 Prediction Based on Best Sale Items

The basic idea for this prediction is to collect the items that have large sale.

For each item, we have one sale corresponding to it. As we have explored in the Data Set and Analysis, 3% of items takes up 50% total sales. So we can say the sales is really unbalanced. This give us a reason to use the top 50% items to predict the purchase.

Four steps to use Top 50% purchase to decide purchase or not:

- Collect the sales of items bought
- Sort the items by its sales
- Picked up the 3% items that will take up 50% total sale and collect them in one set
- If the item is in the set, we predict the user will buy it.

3.2 Prediction Based on Jaccard Similarity

The basic idea of Jaccard similarity approach is to compare the similarity of A and B two items based on the following formula:

$$Jaccard\ similarity = \frac{A \cap B}{A \cup B}$$

where $A \cap B$ = common values of A and B and $A \cup B$ = set of all elements between A and B.

See Figure 2 for better understanding.

Variables	Explanation
A	*all features in set A*
B	*all features in set B*

In project, we enforce Jaccard in following two approaches:

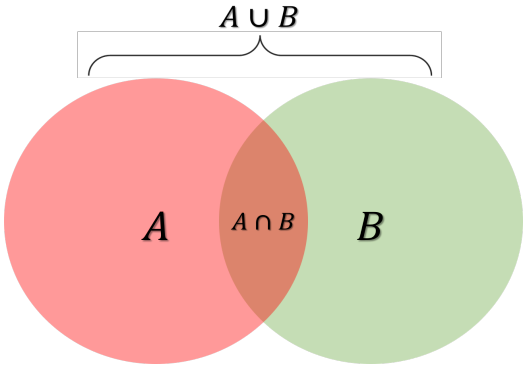

Figure 2: Illustration Relationship of Intersect and Union

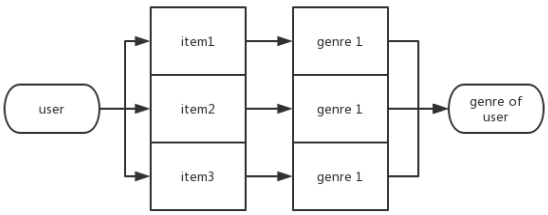

Figure 3: Illustration How to Decide Genre of User

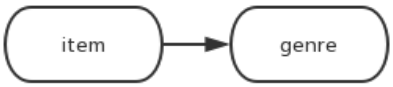

Figure 4: Genre of Items

3.2.1 Applying Jaccard based on the genre that $user_U$ has bought. We compute the similarity between the item genre G_i and the items genre that user has bought G_u.

As we know, most users in the training set have at least bought some items I_u. We extract the genre of these items as user genre $G_u = [gi_1, gi_2, gi_3 ... gi_n]$, as shown in Figure 3.

Also, items itself has a list of genres $G_i = [g_1, g_2 ... g_n]$ (see Figure 4). We compare two lists and output their Jaccard similarity.

3.2.2 Applying Jaccard based on the items that $user_U$ has bought. In this way we generate the Jaccard similarity by comparing the items $user_U$ has bought with items bought by $item_I$ buyers(which means $user_{item_i}$).

When Jaccard similarity exceed threshold T, we predict $p_i,\ i \in n$,that the user will buy this item

$$\begin{cases} p_i = 1, & Jaccard\ similarity > Threshold \\ p_i = 0, & Jaccard\ similarity \leq Threshold \end{cases} \quad i = 1, 2, 3 ... n$$

3.2.3 Evaluation. We evaluate performance of model by calculating its accuracy. Formula as follow has prediction $P = [p_1, p_2, p_3 ... p_n]$ and label $L = [l_1, l_2, l_3 ... l_n]$.

$$Accuracy = \frac{length(P \cap L)}{n}$$

3.3 Prediction Based on Cosine Simiarity

The purpose of cosine similarity $Cos(U, I)$ is to find the similarity between $user_U$ and $item_I$ by calculating average cosine similarity of $user_U$ and users who bought items $U_i = [u_1, u_2, u_3, u_4 ... u_n]$.

$$\mathbf{cos}(U,\ I) = avg\,(\mathbf{cos}(U,\ u_1) + \mathbf{cos}(U,\ u_2) + \ldots\ldots \mathbf{cos}(U,\ u_n))$$

$$\mathbf{cos}(U,\ u_1)\ =\ \frac{U \cdot u_1}{norm(U) norm(u_1)}$$

Then we assume a threshold T to predict p_i, $I \in n$ whether they're similar enough $(Cos(U, I)) > T$ to purchase the item.

$$\begin{cases} p_i = 1, & cos(U, I) > Threshold \\ p_i = 0, & cos(U, I) \leq Threshold \end{cases} \quad i = 1, 2, 3 ... n$$

To apply cos similarity, we need to encode two argument into vector or metrics. To encode user, we apply two approaches:

$$Gen\ type\ encode\ approach = \begin{bmatrix} 1 & 0 \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & 1 \cdots & 0 \end{bmatrix}$$

Gen type encode approach is to describe a user purchase history based on all the gen type for items the user owned by encoding each genre 1 for genre in items user own and encode 0 for genre that aren't in. The approach is short and thus could run faster.

$$Purchase\ history\ encode\ approach = \begin{bmatrix} 1 & 0 & 1 \cdots 0 & 1 & 0 \\ \vdots & & \ddots & & \vdots \\ 1 & 0 & 0 \cdots 0 & 0 & 1 \end{bmatrix}$$

In fact, Purchase history encode approach is precise in describing purchase history by encode each item 1 if user own the item while encode 0 if he doesn't. However, it's longer and takes time to run.

3.3.1 Cold Starter Problem. We define cold starter as those who just start to use Steam $I_u \in \varnothing$. We notice that we can't predict purchase behavior of cold starter as we can't observe his purchase preference in our dataset. Based on the way the model runs, the user always has zero similarity with any items.

3.3.2 Evaluation. Given prediction $P = [p_1, p_2, p_3 ... p_n]$ and label $L = [l_1, l_2, l_3 ... l_n]$, we compute the accuracy rate to evaluate performance of prediction predicted by each model. Accuracy rate is given by:

$$Accuracy = \frac{length(P \cap L)}{n}$$

4 RECOMMENDER SYSTEMS: EXPERIMENTS

We split dataset to train, validation, and test sets by ratio of 50:25:25 which means size of 2,576,605, 1,288,302, 1,288,302 data after shuffle the data. For no purchase observation, we random select users and for each random sample 40 of items he/she doesn't own to fill into validation and test sets until they each have same amount of observation as trainset and have purchase v.s. no purchase data of 50:50.

4.1 Baseline of Prediction

We set baseline as the accuracy of utilizing best sell approach. When predict every user buy the top 3% items, we collect those 3% items into a set and in every observation check if the item in the item set. If it's included, we predict it's a buy but predict it isn't a buy otherwise. When adopt this approach, we find a high accuracy of 0.73, indicating that popular game tendency might be significant.

4.2 Jaccard Similarity

4.2.1 Jaccard based on genre. When we compare the user genre Gu with item genre Gi, from Train dataset we build dictionaries of user v.s. genre, which contain genre of all the items that user bought, and item v.s. genre, which contain genre of the items. Then, we calculate Jaccard similarity of user's genre and item's genre. Afterward, we have the highest accuracy of 0.58 in Test set when setting threshold as 0, which means predict all the users purchase all the items. The result is much lower than baseline and fail to generate valid prediction

4.2.2 Jaccard based on item. We randomly select 1000 of Validation data and then for each datum we compute its Jaccard similarity of items that user bought and items that bought by item i buyer. Utilizing gradient rise method to consider threshold, we find best accuracy of 0.88 when setting threshold as 0.03. The result is better than baseline and do well on predict in both Validation and Test set.

4.3 Cosine Similarity

Encoding gen type of user G_u and gen type of items buyerG_i and calculating cos similarity of G_u and G_i, in each loop make the code run very slow. Therefore, we only random select 30 of items I buyers to calculate cosine similarity and average them. Later, we use gradient rise to determine the best threshold of 0.0001 for the optimal accuracy of 0.63, which is way lower than baseline and we found that gen is not useful in terms of calculate similarity in this case as average purchase per person is 58. It is very likely that each person has bought too many items that they all have games that already cover variety of gen and thus can only find very minor differences.

Running cosine similarity of purchase history is even slower since there are 10978 items in dataset. Thus, while computing cosine similarity we only consider and encode union of two users. This approach accelerates the computation speed 1,000 times. Also, we random select 100,000 out of 2,576,605 Validation data and apply our model to predict it. We eventually predict dataset with threshold 0.07 and collect accuracy of 81% in Validation set and 82% in Test set, outperforming baseline.

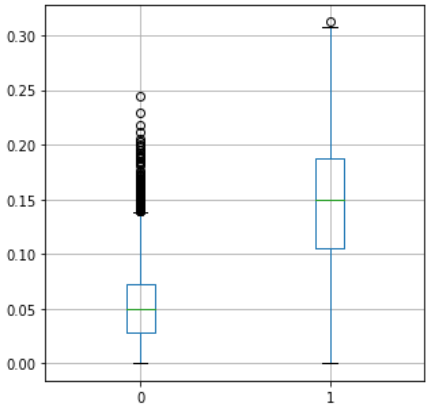

Figure 5: Cosine Similarity difference after plus

As we observed a significant tendency from popular items in Steam dataset, we try to 'plus' score to cosine similarity if user is considering popular items. Using gradient rise to determine 'plus', we eventually find out optimal plus of 0.092144, and the updated model could generate 0.88 accuracy when setting threshold to 0.08500. From Figure 5, given the x-axis is label 0(not buy) or 1(buy) and y-axis is cosine similarity, we find that cosine similarity between buy and don't buy has difference.

5 RECOMMENDER SYSTEMS: CONCLUDE

We have built two successful model. One based on Jaccard similarity based on similarity of item that user bought and items that item buyers bought. similarity. Another is Cosine Similarity by items add score for popular items. They are both robust to cold starter and new items. In fact, Jaccard one's model run faster but only focus on if its similarity or not, while cosines one's model runs a bit slower, but it could collect negative purchase behavior such as refund items.

For further work, we want to apply both models to create a recommender system, making it goes through Jaccard model when excluding refund behavior and goes through cosine similarity when including refund behavior. Afterward it applies text mining model monthly to collect keyword that indicate good sentiment to sustain and bad sentiment to improve.

6 SENTIMENTS ANALYSIS AND TF-IDF PREDICTIONS

6.1 Dataset Overview

We use Steam Video Game and Bundle Data credited by Julian McAuley[4] for training purposes. See Table 5 for summary of data structure.

6.2 Sentiment Analysis of Customer Reviews

Sentiment analysis has been proved to be a valuable technique of text mining in the application of recommender system. In this case

[4]The existing Bibstrip data, copyright text and permission block in the sample file are dummy values, so the user needs to provide the correct values required for the submission in the metadata dialog box.

Table 2: Steam Video Game reviews data

Total:	59,305
Positive:	52,473
Negative:	6,832

Figure 6: Positive Sentiment Words

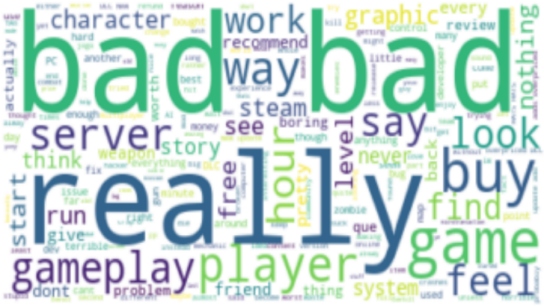

Figure 7: Negative Sentiment Words

of steam video game, players are able to provide text reviews, comments or feedback to the items they have purchased, from which sentiments, opinions on item features can be extracted. Before starting our prediction task based on the text reviews of customers, we have to take a glimpse of how satisfying items are informed by positive sentiment words and how unsatisfying items are informed by negative sentiment words.

To distinguish positive sentiment words from negative sentiment words, first of all we are supposed to construct two dictionaries, one of which stores reviews that indicate positive sentiments while the other of which stores ones that indicate negative sentiments. After doing so, we adopt word cloud visualization to vividly illustrate our results[2].

From the word cloud shown above, we can tell that strongly positive words such as 'great', 'good', 'amazing', 'fun' etc. typically appear in reviews that recommended items received while typically negative words such as 'bad', 'boring', 'problem' can be found from reviews that unrecommended items received. This, to some extent, reveals that we are likely to collect some evidence about if an item is a popular one or not by taking a look at the comments that are left by customers who have purchased it.

Table 3: Top Words Based on TF-IDF

Item ID	Top Feature Words	TF-IDF
43110	ammunition	7.8522
	stealth	7.7874
	light	7.1468
	shooter	7.0829
	horror	6.0367
730	pew	1066.7267
	csgo	413.1349
	cs	277.8320
	game	263.3887
	strike	204.5536
...	...	...

Furthermore, we can even figure out which words are typical of each item by calculating TF-IDF of the 1000 most common words extracted from all reviews left by customers. To achieve that goal, we have to follow the steps below:

- Collect all text reviews from the dataset and construct a list to store 1000 most frequently-appeared words.
- After obtaining the most common words, we are supposed to calculate the TF-IDF of each word for each item's reviews.
- Rank the words for each item in terms of the value of TF-IDF and take the first 5 as the symbol words that may represent the latent reasons why the item has been recommended by customers or not.

A part of results are shown as following **Table 3**:

7 HELPFUL RATE PREDICTION: BASED ON TF-IDF

7.1 TF-IDF Introduction

TF-IDF (term frequency – inverse document frequency) is a commonly used weighted technology for information retrieval and data mining.[5] To evaluate the importance of a word to a document set or to one of the documents in a corpus. The importance of a word increases directly with the number of times it appears in the document, but at the same time it decreases inversely with the frequency of its appearance in the corpus. TF-IDF weighted forms are often used by search engines as measures or ratings of the degree of correlation between files and user queries. Based on TF-IDF, we can figure out the importance of each word and pick up the useful work from the review and use linear model to analysis the latent influence of other features.[3]

7.2 Dataset Exploration

When we explore the user review dataset and the steam review system, we find that the review of user are displayed based on the helpful rate(useful amount divided by total amount).And those with a high helpful rate are considered as a good review. In the steam community, review is considered as one of the most essential reasons people buy games. So keep a good and objective rank of the review area is quiet important for a game and for steam company. More reasonable rank can increase people's willing to buy a game and increase the income of steam company.

But in the dataset, we find that there are also 28,215 reviews without helpful or not statistics. And there are a lot of useful information in the reviews and some of them should be displayed in the front. They have no helpful or helpless definition They don't have the definition because they always appear in the last part of the comments section, rather than because the comments aren't useful and people don't want to do something with it. So many users don't have the opportunity to see the comments and therefore can't give a definition of whether they're good or bad. So to maximum the power of all the reviews, we need to give a helpful rate score to those 28,215 reviews and rank all the review for a specific game again. To fill in all the blank in the helpful part, follow these steps :

7.3 Model Establishment

- Make an item and review dictionary for the whole dataset. This is used for the final ranking for all the review and make sure those ranking review can apply accurately back to a specific item.
- Separate all the reviews into train set, validation set and test set. Test set of course should be all the review without helpful rate and the training set and validation set are randomly selecting from the other part. Considering the fact that those reviews with a low total helpful or not helpful amount may be not so objective. We will set a variable here to set a burden to how many helpful or not helpful mark can go into our training model and finally we will change this variable to get a best MSE in the validation set.
- We calculate the TF and IDF in the train set and get the TF-IDF list for each review.
- Applying the TF-IDF list and consider it as x. And then we consider helpful rate(user think the review is helpful amount divided total comment amount) as Y. Applying the X and Y into linear regression model and using the validation to choose the best regularization parameter.
- We start to mine the train set and validation set using lower character and skip those punctuation to get a cleaner test. We apply 8 kinds of method including bigram or unigram, keep punctuation or skip punctuation, using words count or TF-IDF.

7.4 Evaluation

We find that when we use 5 total help amount as the minimum border of total helpful amout for the train and validation set and using unigram TF-IDF method after skipping the punctuation we get a very good final result. The regularization parameter is 100 and the best MSE in our model is 0.06(significantly increased compared to baseline 0.20). It means that we can predict the helpful rate of one specific review and we can apply those review to the item they belong to. Finally, we use sort function to rank the review for the item again and steam company can display the review based on the ranking and get a better influence.

8 ITEM-SCORE PREDICTION

In the user-review dataset, we can find a binary category named recommend or not and one recommend is attached with a relative review. In the whole dataset of steam community, we cannot locate a specific rating system for a specific item or we call it game. Therefore, contingent on the method of TF-IDF, we can build up a model to predict how likely a user is to recommend an item based on their reviews. Although the dependent variable 'recommend' is present as a binary output, we can convert it to a continuous one by figuring out the percentage of users who actually recommended a certain item. After assigning the value of percentage to the response variable and constructing a feature matrix comprised of vectors representing TF-IDF of the 1000 most important words for each item, we are able to adopt the approach of linear regression to make predictions. To predict the item score, we are supposed to follow these steps :

8.1 Model Establishment

- In addition to constructing an item-review dictionary, we also have to establish an item-probability dictionary to measure the likelihood that a certain item might be recommended by users, which can be calculated as the percentage of users who recommended the item among all users who purchased it .
- Having obtained the item-probability dictionary, we are enabled to build our regression model by assigning the probability that a user might recommend an item to the response variable and featuring the independent variables with values of TF-IDF for the 1000 most important words extracted from the reviews of users. Besides, we incorporate regularizers ranging from 0.01 to 100 into our model in order to avoid overfitting problem by penalizing model complexity.
- To procure the final predicted score, we multiply the predicted probability of recommendation for each item by 5 as higher probability indicates that a user are more likely to give high ratings to a certain item.

8.2 Evaluation

To evaluate the validity of our predictions, we implement a train-validation pipeline (the proportion of train set to validation set is 5:1) to check out the performance of our model by examining the MSE on the validation set with the best value of λ. The results show that the best λ for our model is 10 and its MSE on the validation set is 0.2896, which is a sweet-pot point of all values selected in our λ set.

9 SENTIMENTS ANALYSIS AND TF-IDFPREDICTIONS: CONCLUDE

After mining the review dataset and using TF-IDF to predict ,we mainly make three good model and good application of them. From the sentiment analysis ,we use words counts sentiment analysis and work cloud method to show the top 5 and bottom 5 features word for a specific game. That can help steam company to define why the game is popular or unwelcome.

In the helpful rates prediction part, we build a high accuracy model to predict the helpful rate of unmarked review, which can help steam company to display all the review reasonably and effectively.

For the item-score predictions, we build a model to get the latent score from user's review and get a good model. We can We can use this hidden score as a comparison of actual scores and consider it as a reference score for the game frame.

10 OVERALL CONCLUSION

Based on the steam dataset, we make two recommender system prediction model with high accuracy and two useful sentiment analysis model. These model can help companies get a more accurate definition and position of their games and users, thus increasing sales and selling more games. Our recommendation system can recommend bundles that users are more likely to buy, while a review helping rating system can improve the likelihood that users will buy games by effectively ranking reviews. So all of our effort can help the company achieve higher success in business.

For the future work, we can combine the sentiment analysis and recommender system together and use the sentiment words in our model to modify recommender system model to get higher accuracy.

REFERENCES

[1] Christopher M Bishop. [n. d.]. PATTERN RECOGNITION AND MACHINE LEARNING. ([n. d.]).
[2] Weiwei Cui, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X Zhou, and Huamin Qu. 2010. Context preserving dynamic word cloud visualization. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific*. IEEE, 121–128.
[3] Dean P Foster, Mark Liberman, and Robert A Stine. 2013. Featurizing text: Converting text into predictors for regression analysis. *The Wharton School of the University of Pennsylvania, Philadelphia, PA* (2013).
[4] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and Personalizing Bundle Recommendations on Steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1073–1076.
[5] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation* 60, 5 (2004), 503–520.