Recommender System for Publisher of Technology News

Yuzi Luo
Stanford University
yuziluo@stanford.edu

Yue Li
Stanford University
yueli96@stanford.edu

Yawen Sun
Stanford University
yawensun@stanford.edu

Advisor:
Rok Sosic (rok@cs.stanford.edu) and Jeffrey Ullman(ullman@gmail.com)

Abstract

Digital Trends (http:www.digitaltrends.com) is a large online technology news provider with over 100 million monthly users. To attract more users and achieve longer interaction with each user, as well as exploring smarter ways for ads placement, it is essential for the website to provide a content personalization service. In this project, we performed data analysis on users' interactions with the website over 15 months. We aimed at exploring users' behavior, interest and methods for providing personalized content for each user. In the end, we built a news recommendation system for the website through similarity learning and a Wide and Deep learning model. We evaluated our model through Mean Reciprocal Rank (MRR). Results show the proposed method is promising for the website's future content personalization service. Source code can be found on our Github.[1]

1 Problem Description

Digital Trends (http:www.digitaltrends.com) is a technology news website that provides news, reviews, guides, etc, about technology and consumer electronics products. With over 40 million users from across the world monthly visiting the website, how to lengthen each users' interaction with the website is an essential question to think about. Users visit the website to read more about trending technology and product of interests. However, the huge amount of information published online makes it hard for users to navigate the content they are interested in. Thus, to maximize this group of users' engagement, we are going to optimize the reading suggestions shown to them in a personalized way. Content personalization also helps with ads placement that are most likely to receive a positive response from each particular user. The goal of this project is to build news recommendation systems and provide personalized content for each user on the Digital Trend Website.

To build a recommendation system, there are several potential challenges to think about. The first is how we are going to build and maintain user profiles that describe users' reading preference over time as well as news articles' profiles that reflect the content and similarities among news articles. In this project, we need to consider what kind of information reflects users' interest and if there are other factors that may affect users' preference. For example, for a recommendation system, especially news recommendations, the recency of news and short-time big news event are important factors that affect users' choices. That means we should make recommendations that are highly related to the current trend. Also, users' preference of interests are also changing over time and are influenced by big news events. When generating recommendations, we should keep a balance between their long-term and short-term preferences. Another challenge with a recommendation system is the idea of exploration and exploitation. That is, we can not keep recommending content similar to what the user has viewed before. Exploration of different content that may attract a user for longer interaction with the website is also essential for a good recommendation system model. Besides, we should also deal with the cold start problem which refers to the situations where little information is known about the preference of a user.

2 Related Work

Recommendation systems are often classified into two main categories: collaborative filtering (CF) and content-based filtering (CB) [4]. Content-based filtering is based on analyzing a reader's past interest to recommend similar items. Collaborative filtering relies on the interest patterns of a community. However, according to [4], many research papers use a hybrid approach and combine content-based filtering and collaborative filtering. This suggests that the combination of content-based user profiles with patterns in a community is considered to be promising.

We explored some papers for google news recommendation. [3] used CF with MinHash Clustering for google news recommendation. Later in 2010, [5] extends the previous work by considering reading patterns in the community. They proposed a Bayesian framework

[1] `https://github.com/yokostan/Recommender-System-for-Publisher-of-Technical-News`

to remove the community bias from the user's clicks in each topic to extract their long term interest. During prediction, they also take each user's short term interest into account, which is highly related to those big news events. Different from previous work, [2] proposed a machine learning model to solve this problem. They combine a linear model and a neural network for a balance between exploration and exploitation, which also achieves great performance in their live experiment.

3 Dataset Overview

Thanks to Digital Trends, who provided us with two main datasets useful for this project. The first one is a Content History Dataset, which records information about 170,000 news articles published on their website. This dataset is collected through WordPress, which is a content management system based on PHP & MySQL. The collected information includes each news article's unique id, publish date, categories, title, tags and url. They also provided us the Event Log Dataset, which contains all users' interaction with the website over 15 months. This dataset is collected through Snowplow, which is a real-time event data collection platform. It records some useful information for this project including each users' unique id, location, page url that he/she clicked on, and timestamp of this event. Below we gave several examples of these two datasets and information we extracted for our proposed methods.

- Content history

Table I: Content History from WordPress

News ID	Publish Date	Category & Tags	News Title	News URL	...
1002075	4/3/2019	fashion|travel|Travel |Travel|packing|luggage	5 Best Travel Clothes Brands You Can Get Away	www.themanual.com/ travel/best-travel-clothes-brands	...
396726	4/2/2019	computing|buying-guides|Computing|hdd|ssd	SSD vs. HDD	www.digitaltrends.com /computing/ssd-vs-hdd	...
1000917	3/29/2019	fashion|Fashion & Style |leisure|exercise|fitness	7 Track Suits to Take You Way Beyond The Finish Line	www.themanual.com/ fashion/best-track-suits-for-men	...

- Events log (670 GB)

Table II: Event Log from Snowplow

Domain User ID	Event Timestamp	Location	Clicked URL	...
e09577f8-0bbb-477c-b3ad-82754fc52c7e	1/29/2018 1:22:56 PM	US	https://www.digitaltrends.com/smart-home-reviews/netgear-arlo-pro-2-review/	...
c895f207-6cc8-4ca2-8eb6-c377ff25d756	1/29/2018 1:22:56 PM	ZW	https://es.digitaltrends.com/android/como-rastrear-un-celular/	...

One problem with this dataset is that given the fact that Digital Trends Website doesn't have a sign-up system, snowplow does not give each user a unique id for long term tracking. However, it does record the user IP address and the domain user id. Since a single user can have many IP addresses and many users can have the same IP address if they are in the same office or lab, it is not a good idea to use the IP address for user identification. The domain user id is generated by the Javascript tracker and stored in a first party cookie. The shortage of using the domain user id for identification is that if a user deletes his cookies or the cookie expires, or the user visits from another browser or computer, a new domain user id will be generated and we may lose track of their activity. However the good news it that through domain user id, we can make sure that all event logs with the same domain user id came from an unique user, which is essential for analyzing his/her interest and behavior.

Another problem is how to combine these two tables. From the examples above we can see the only way to combine them is through web page URL. Since each news article has a unique url, we can use this to combine these two tables and find out what content a user has clicked on. This can give us the information of a user's interest.

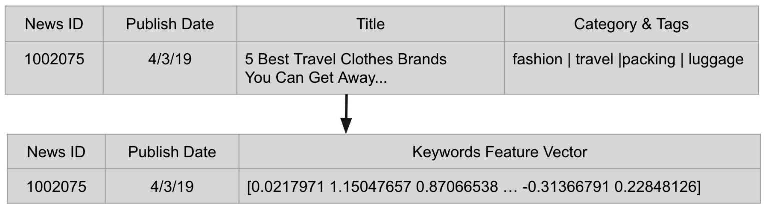
4 Methodology

4.1 Data Preprocessing

As mentioned above, one challenge of this project is to create profiles for each user and news article. Users' profiles reflect their viewing history and potential interests. Similar user profiles indicate that they have similar interests and are likely to read the same news article. Also, a news article's profile is a reflection of its content. It can also indicate similarities between news articles. In this project, we use feature vectors to represent user and news article's profile. Below we described our methods for feature extraction in detail.

4.1.1 Posts Features Extraction

For posts' features extraction, as Figure 1 shows, we extracted keywords from tags and titles for each news article in the Content History Dataset, turned each keyword into a 32 dimension feature vector through Word2Vec, and calculated the average for the final content feature vector representation. This feature vector reflects the content of each news article. We can say that vectors of posts with similar keywords point to close directions in the high dimensional space. However, one shortage of this method is that keywords have a very limit representation of the content. Sometimes news article's title can not well reflect the actual content. And articles with the same tags may also have very different content. This is, in fact, a limitation of our current dataset. For now, we just proceeded the project with the current dataset. But it would be better if we have the actual content for feature extraction.



News ID	Publish Date	Title	Category & Tags
1002075	4/3/19	5 Best Travel Clothes Brands You Can Get Away...	fashion | travel |packing | luggage

News ID	Publish Date	Keywords Feature Vector
1002075	4/3/19	[0.0217971 1.15047657 0.87066538 … -0.31366791 0.22848126]

Figure 1: Posts' Feature Extraction

At initial attempt, we split posts' tags with delimiter '|' and title with delimiter ' '. These split words from tags and titles serve as the keywords to represent each post's general idea. Then we conducted 'word2vec' on these split keywords and averaged the value of the word vectors among each post, respectively. For our final feature vector of posts, we assigned weights to posts_tags and posts_titles and outputted a combined 32-dimensional word vector. The formula of our word vector generation is as below:

$$f_{post} = w_{tag} f_{tag} + w_{title} f_{title} \tag{1}$$

where f_{post} is the total feature word vector, f_{tag} and f_{title} are the word vectors for tags and titles respectively and w_{tag} and w_{title} are the weights of the word vectors. Here, we empirically set $w_{tag} = 0.7$ and $w_{title} = 0.3$.

After an overview of our to-be-mentioned clusters based on the above word vectors, we discovered some optimization to Digital Trends posts. Firstly, there are many general tags that appeared to be the top ten frequent tags among over half of all the clusters we have. Thus, we eliminated general tags like news, web, mobile and computing. Also, there are some blank tags assigned and many '' marks from tags having 'ab' format. So we eliminated ' ' and '' from tags as well. For titles, we find some common patterns and highly frequent words that are less meaningful in differentiating between posts in Digital Trends posts, such as 'everything you need to know' pattern and words like 'best' and 'new'. Accordingly, we added more stop words unique to Digital Trends besides the original nltk stopwords package. We also took Spanish stopwords into consideration for feature vectors optimization

4.1.2 Posts Clustering

After extracting feature vectors from each post, we performed K-Means clustering on these feature vectors. In this way, we were able to generate candidate posts whenever a recommendation request is sent to our system. Also, we could use posts clusters for user feature extraction, which we will explain in the next section.

Here we used K-Means Clustering and Elbow Method to select the optimal number of clusters to use. For distance measurement in K-means clustering, we normalized the feature vectors and calculated the cosine similarity. In Elbow Method, we first plotted the relationship between distortions, which is the sum of squared distances from each point to its assigned center, and the number of clusters. Then we chose the point where the marginal gain dropped dramatically and gave an angle in the graph, for the optimal number of clusters which balances between computational complexity and clustering performance [1].

Firstly, we explored the range where the number of clusters is below 200. From the elbow graphs Figure 2 and Figure 3, we figured out that for both our initial and optimized versions of word vectors, the optimal number of clusters is around 50. The distortions when we have 50 clusters is around 0.3 and 0.2, respectively. This means that our optimization approaches effectively reduced the sum of distances within the clusters and have yielded clusters with more consistency.

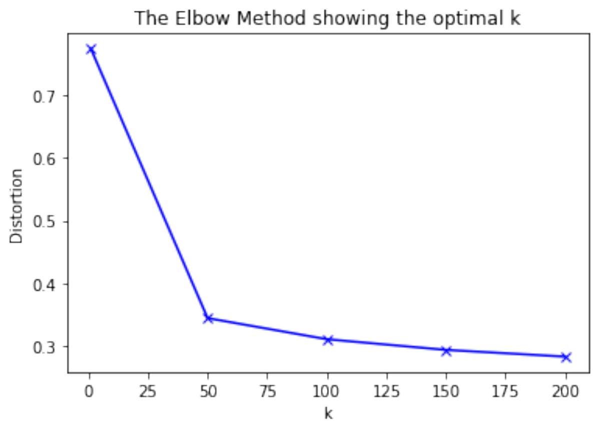


Figure 2: Elbow graph for K-Means distortions of initial vectors

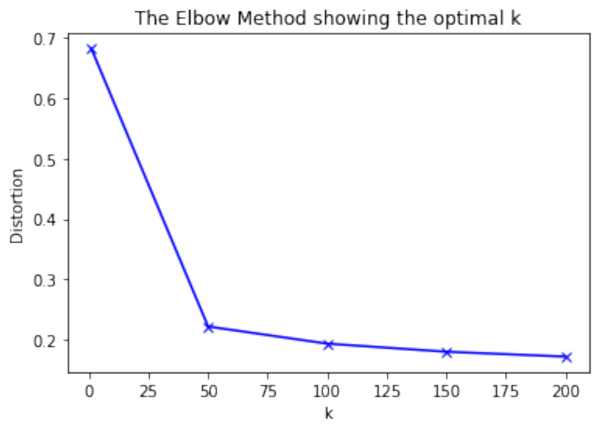


Figure 3: Elbow graph for K-Means distortions of optimized vectors

Even for the 50 clusters of word vectors before optimization, we were able to get high semantic consistency for the clusters. After examining the highest frequency titles and tags of all 50 clusters, we have summarized the topic for each cluster as in Table III.

Table III: 50 clusters from unoptimized word vectors

TV shows/Nintendo Switch	Iphone/App/Android	3D Printer/Drone/Tech product
TV show/Movie/Play	How to/The best	Car(brands)
Home Appliances(kitchenware)	News and rumors	Android
Games with Weapons	Spanish social media	Menswear and Gear
AT&/5G	Photos/Photoshop	Smartwatch/Smartwallet/Fitness tracker
Sling TV/Netflix/DVD CD	Furniture	AMD/Intel/Chips/CPU
Spanish vehicles	Car supplies	Software setting
Bitcoin	Scifi	Google Home Mini and Amazon Echo
Spanish digital products	Windows/Computer tutorial	Spanish smart phones
Men's cosmetic	Samsung Galaxy Note/S	Headphones
Cars(speed)	Laptop	Digital Trends Live/DT Daily
Shopping	Recipes	bike/speeding pad/yoga/wood chopping
Home improvement(kitchen and bedroom)	Cameras	Spanish/English
Music Streaming	Aggressive Opinions	Flight/Tourist Attractions
Video Games	VR	XBOX
Travel	MacBook	Bargain
Nintendo/NES	Apple products	

As the distortions when the number of clusters is 200 are around 0.2 and there still seems to be some potential to further decrease, we extended our exploration range and examined various properties of the clusters.

We firstly looked at the size distribution of the clusters. Table VI shows the minimum size, maximum size and average size of the clusters when the number of clusters equals 50, 100, 200, 400 and 1000. From the table, we can conclude that the minimum size, maximum size and average size of the clusters all decrease when the number of clusters increases from 50 to 1000. When the number of clusters is 1000, the minimum cluster size is only 6, meaning there are only 6 posts in the smallest cluster, which is relatively too small a cluster when we have over 170000 posts to recommend in total. Thus, we confined our range from 50 to 400.

Table IV: Size Distribution of Clusters

number of clusters	minimum size	maximum size	average size
50	656	16389	3527
100	389	9959	1763
200	79	5741	882
400	57	3835	441
1000	6	2417	176

Figure 4 shows the size distribution of the clusters when their total numbers are 50, 100, 200, 400 in yellow, grey blue and orange respectively. As stated above, the overall size of the clusters is negatively proportional to the number of clusters.

Although we have filtered out the cases when the number of clusters is above 1000 due to cluster imbalance, we are still curious about where the lowest distortion when the number of clusters equal to 200 in Figure 2 and Figure 3 falls in the bigger picture. So we examined the distortions of the clusters up to the range of 10000 clusters. As Figure 5 show, the 200-cluster distortion it is still relatively high compared to the distortion when number of clusters is 10000. The general case is that as we increase the number of clusters, the distortion is always going to drop down for we are supposed to get relatively smaller and more consistent clusters.

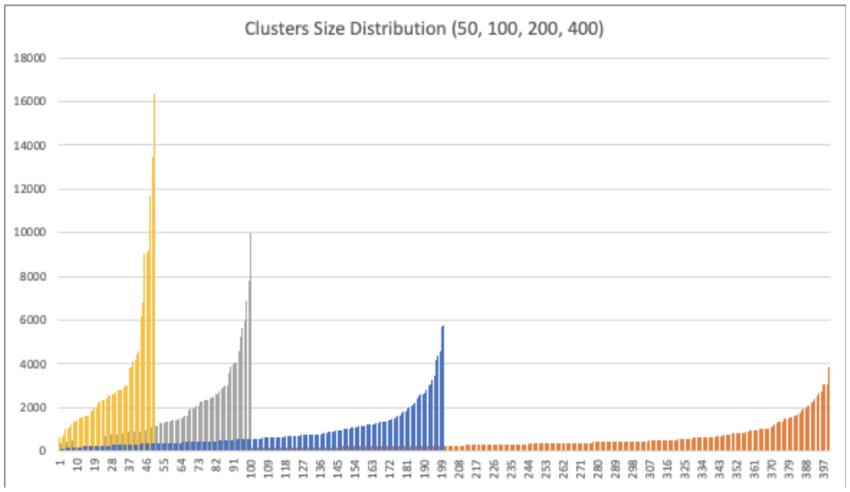


Figure 4: Cluster Size Distribution(number of clusters 50, 100, 200, 400)

Table IV, Figure 4 and Figure 5 combined, we kept number of clusters primarily to be 50, 100, 200 and 400. Examination of the semantic meaning of each clusters provided us with new thoughts on cluster optimization.

Here we presented two examples of our clusters: alcohol cluster and gaming cluster.

As the number of clusters goes up from 50 to 100, both clusters are split to smaller sub-clusters compared with previous ones, with their most frequent title words generally the same.

For the alcohol cluster as Figure 6 shows, when the numbers of clusters are 50 and 100, the most frequent title words are beer, cocktail, whiskey, drink, wine, food and recipe and the size of the clusters remain stable around 2770. While when the number of clusters is raised to 200, the one big alcohol cluster is split to one relatively big cluster of size 2685 with the same most frequent title words and one relatively small one of size 168 having bacon, booze, chef and feasting as its most frequent title words. When the number of clusters goes up to 200, the clustering result remains roughly the same, with few posts clustered as a distinct small group. As the sizes of these two clusters vary a lot from each other, the posts in the relatively small cluster can be viewed as outliers of the original bigger alcohol cluster.

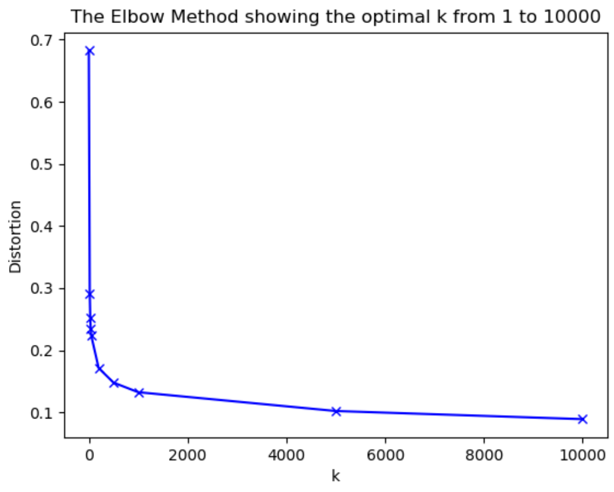


Figure 5: Elbow graph for K-Means distortions of optimized vectors

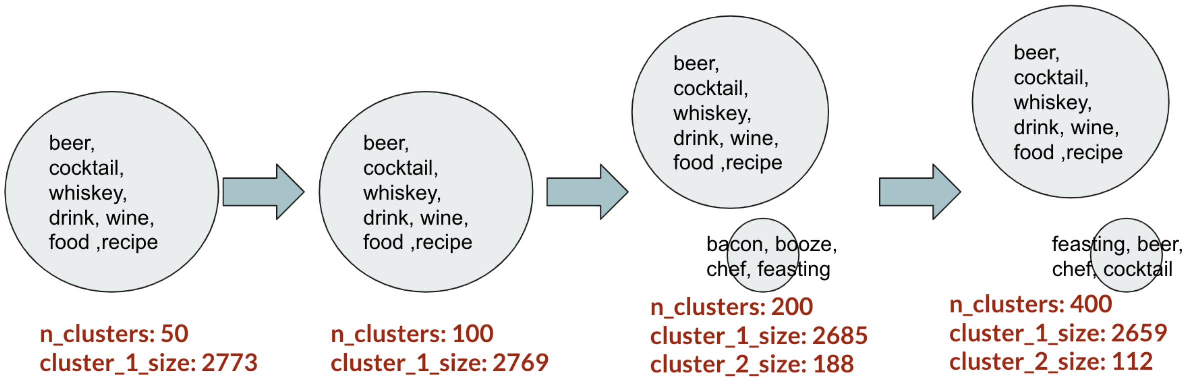

beer, cocktail, whiskey, drink, wine, food ,recipe
n_clusters: 50
cluster_1_size: 2773
beer, cocktail, whiskey, drink, wine, food ,recipe
n_clusters: 100
cluster_1_size: 2769
beer, cocktail, whiskey, drink, wine, food ,recipe
bacon, booze, chef, feasting
n_clusters: 200
cluster_1_size: 2685
cluster_2_size: 188
beer, cocktail, whiskey, drink, wine, food ,recipe
feasting, beer, chef, cocktail
n_clusters: 400
cluster_1_size: 2659
cluster_2_size: 112

Figure 6: Top Frequent Title Words for Alcohol Cluster

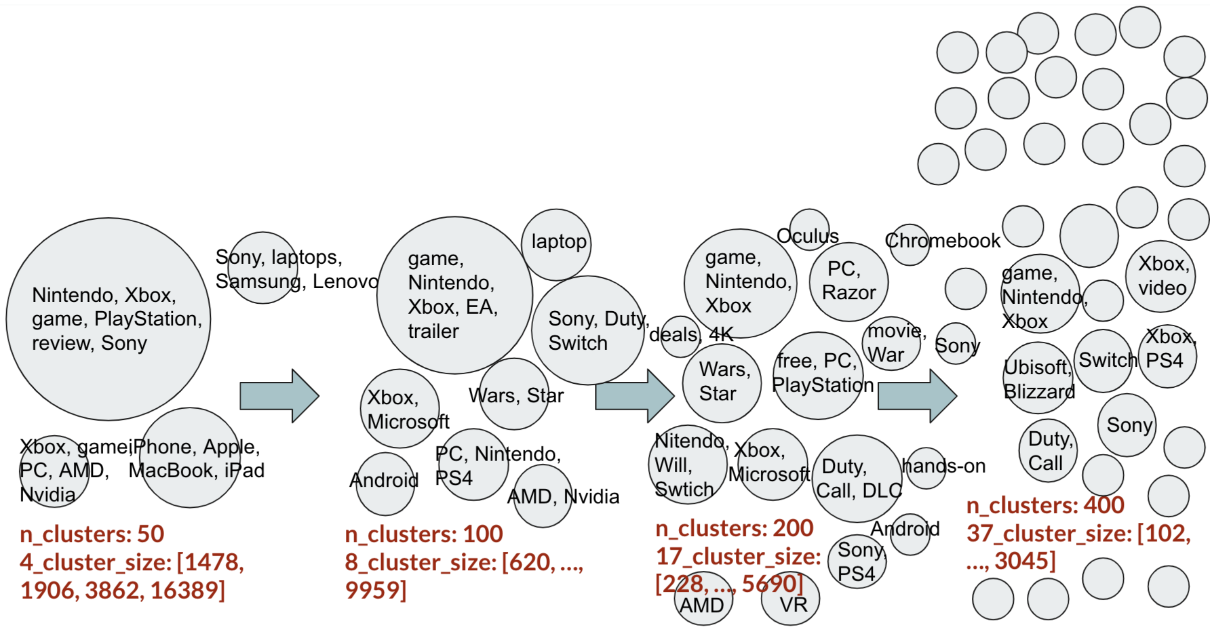
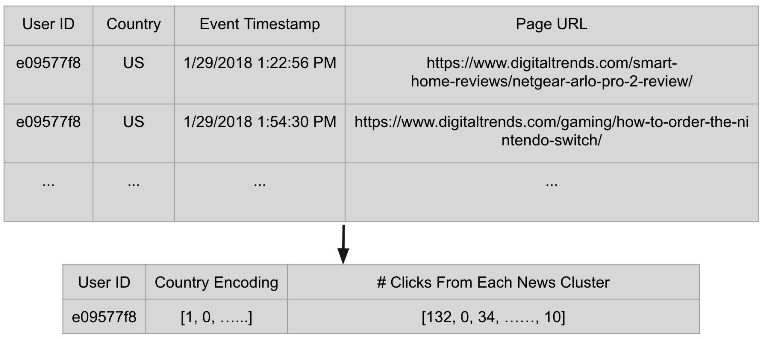


Figure 7: Elbow graph for K-Means distortions of optimized vectors

As Figure 7 shows, for the gaming cluster with larger size and more sub-categories within itself and can be split more diversely when the number of clusters gets bigger. When the number of clusters is 50, 100, 200 and 400, there are 4, 8, 17 and 37 gaming clusters respectively. As more sub-categories are split from the original larger clusters, the variations of the content began to show. As the number of clusters went up, the size of the largest major gaming cluster went down while more clusters with specific topics merged, such as the Star Wars cluster, Duty Calls cluster and AMD cluster.

4.1.3 Users Feature Extraction

For users feature extraction, three factors that could reflect each user's preference are considered: the user's location, the news articles the user has viewed before, the time that the user viewed the article. We use binary encoding to represent the user's location. We then calculate his/her number of clicks from each cluster. Take users' reading preference over time and news recency into consideration, we calculate the difference of the time the user viewed the article and the article's publish time. These features are combined together to form a feature vector for each user, as shown in Figure 8.



User ID	Country	Event Timestamp	Page URL
e09577f8	US	1/29/2018 1:22:56 PM	https://www.digitaltrends.com/smart-home-reviews/netgear-arlo-pro-2-review/
e09577f8	US	1/29/2018 1:54:30 PM	https://www.digitaltrends.com/gaming/how-to-order-the-nintendo-switch/
...	...	...	...

User ID	Country Encoding	# Clicks From Each News Cluster
e09577f8	[1, 0, ...]	[132, 0, 34, ..., 10]

Figure 8: Users' Feature Extraction

4.2 Learning Models

4.2.1 Similarity Learning

The similarity learning model is a simple linear model we use as our baseline. In this model, we aim at learning a matrix W that parameterize the similarity function:

$$f_W(u, p) = u^T W p \tag{2}$$

where u represents a user and p represents a news article. The similarity function gives us a value between 0 and 1, which tells us the likelihood of a user clicking on a specific post. We then calculated the binary cross entropy loss for updating the weighting matrix W. The loss function is defined as:

$$L_i = -y \log(f_W) - (1 - y) \log(1 - f_W) \tag{3}$$

where label $y = 1$ if a user has clicked on the post, $y = 0$ otherwise.

4.2.2 Wide and Deep

The wide and Deep model is also a popular news recommendation model proposed in [2]. It jointly trained a wide linear model and a deep neural network. As described in [2], the wide component is good at memorization feature interactions and recommend closely related items. As mentioned before, in a recommendation system, we can not keep recommending similar items since users may soon get bored and lose interest. It is important to explore users' potential interest and generalize our recommendation. Since generalization requires more feature engineering effort, we need a deep neural network that explores unseen feature combinations.

The wide component is a linear model of the form $y = w^T x + b$. Here y is a probability between 0 and 1 that reflects the likelihood of future click between an user, post pair. The deep component is a neural network with each hidden layer of the form

$$a^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)}) \tag{4}$$

where l is the layer number and f is the RELU activation function. $W^{(l)}, b^{(l)}, a^{(l)}$ are weights, bias and activation at the l-th layer. The model's prediction is represented as

$$p = \sigma(w_{wide}^T x + b + w_{deep}^T a^{(l_f)}) \tag{5}$$

where $\sigma()$ is the sigmoid function, w_{wide} is the weight of the wide model, b is the bias term, and w_{deep} is the weight applied on the final activation $a^{(l_f)}$ of the deep model. We also use binary cross entropy loss that defined in equation (3).

5 Experiments

5.1 Setup

In the experiment, we chose the first 9 months' click events for each user as the training set and the last 6 months' clicks as validation and test set. For validation and test, we used half of the users for validation, and the other half for test. We built users' profiles based on data in the first 9 months, and use these profiles to predict users' future clicks. To do this, we need to select users that had a long-term interaction with the website over 15 months so that each of them had clicks history in both training and test set. After selection, there were 8425 users left, each of them had at least 10 clicks in both training and test set. Also, we noticed that some countries had very few users, which might not be representative. Thus we also filtered out users from those countries that had less than 50 users interacting with the website. After preprocessing, there were 7123 users left from 14 countries.

Among those 7123 users, there were approximately 3,500,000 click events, each click event is a positive data point containing features of a user and post pair. This resulted in a very large dataset. We first randomly sampled 5% of the total positive data points which are around 170,000, and get an initial result of our two models using cluster number 50. Then we used the whole dataset to train both models using cluster number 50, 100, 200 and 400.

One issue with the experiment is to generate negative data points. In fact, one limitation of our dataset is that we only have users' click information. However, if there were 25 news articles presented on a web page, knowing what were the other 24 articles that a user did not click on is also very important because they serve as the negative data points in our learning model. However, since we did not know have information, the way we did it was to choose articles that published at the same time as the one user clicked on as our negative data points. Given the fact that the website itself tends to put the newest articles on top of the web page, We assumed that most articles presented to users on one page were published around the same time.

In prediction, we generated a list of candidate news articles for each user. Those articles came from the clusters that the user has clicked on. This can reduce the computational cost and gives each user a recommendation more quickly.

5.2 Evaluation Method

Recommender systems are mostly evaluated in the following three approaches: offline experimentation and simulation based on historical data, laboratory studies or A/B tests on real-world websites[4]. In this project, we used offline experimentation to measure the performance of our model. For evaluation, we use mean reciprocal rank (MRR) as a measure of the performance:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{6}$$

MRR is the average of best reciprocal ranks. To be more specific, for each user, we had a list of candidate posts. Our model then gave each candidate post a score that represented the likelihood of the user clicking on this post. Then we chose the highest rank of a good guess post according to the posts the user clicked in the validation or test set, and set it as the best rank for this user. Here are two approaches to define a good guess post:

- as the exact post the user clicked in the validation or test set.
- as a similar post to one post the user clicked in the validation and test set.

The first approach is natural and does not need further explanation. The second approach is reasonable in our scenario. In reality, even though the user did not click on one post, name it post A, that is similar to the post, name it post B, that he/she actually clicked in the validation or test set, he/she was likely to click on post A if it was shown on the website instead of post B, based on the assumption that the user was interested in the content of post B at that moment, and post A and post B have similar content. So if we recommended post A instead of post B, the user would click on post A with high probability.

MRR is the average of all users' best reciprocal ranks. As we can see, MRR is a value between 0 and 1. The larger the MRR, the better our model's prediction.

5.3 Results

5.3.1 Subsampled Dataset

We first evaluated our method with subsampled data points with cluster number 50, and evaluated the MRR of the highest rank of the exact post. Table V shows the MRR of these two methods on validation and test set. Figure 9 and 10 show the distribution of test set's best ranks of two models. We can see although the MRR is relatively low, wide and deep model do shows better performance than our baseline model. This suggests that the result can be further improved by tuning hyperparameters and performing the learning process on the whole dataset. Another idea we have from this result is that due to several limitations of our dataset and perhaps the limitation of our method itself, it is really hard to give highest scores to the exact same posts user has clicked in the val and test set. Our idea is that we can improve the evaluation method so that once our prediction is really similar to the target, we should say it is a good prediction and give it a higher evaluation score.

Table V: MRR of Exact Posts for the Two Learning Models, Cluster Number = 50

	Similarity Learning	Wide and Deep
MRR on Val	0.00117	0.00496
MRR on Test	0.00116	0.00504

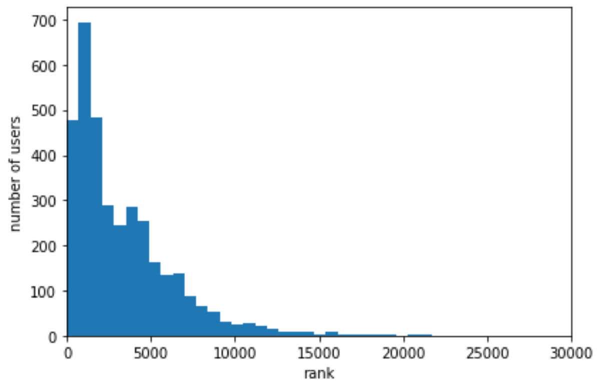


Figure 9: Similarity Learning: Number of Users v.s. Best Rank

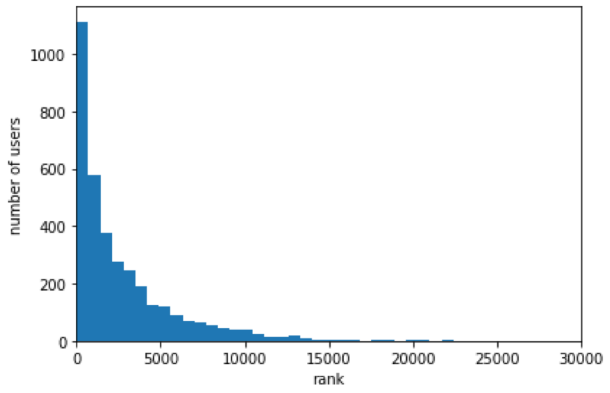


Figure 10: Wide and Deep: Number of Users v.s. Best Rank

5.3.2 Whole Dataset

We then evaluated our method with the whole dataset with cluster number 50, and evaluated the MRR of the highest rank using both approaches. Table VI shows the MRR of these two methods on validation and test set. Figure 11 and 12 show the distribution of the best ranks of the wide and deep model of both approaches on the test set. When evaluating on MRR by exact posts, there is an increase in the MRR of both models, which means using the whole dataset improves the performance of the learning process. However, the MRR is still pretty low. From Figure.11, we can see there are only 1/3 users got the best rank higher than 1000, which seems that our model is generating total random predictions. However, it is actually not the case. Due to the limit information we have for each news article, the similarity among articles within the same cluster is actually pretty high, which makes it hard for us to predict the exact same posts as the ground truth. We realized that we may need a better way to evaluate our model.

We then modified our evaluation method and say that a prediction is good enough when our prediction and the ground truth are within the same cluster, and have a cosine similarity higher than 0.95, which is the average similarity of among all the clusters. When evaluating with this method, we see from Table VI and Figure 12 that the MRR of both models increased a lot. And about 2/3 of users got a best rank higher than 10. This suggests that our model is actually good at generating high similarity post as the ground truth.

Table VI: MRR for the Two Learning Models, Cluster Number = 50

	by Exact Posts		by Similar Posts, Threshold = 0.95	
	Similarity Learning	Wide and Deep	Similarity Learning	Wide and Deep
MRR on Val	0.0012	0.023	0.0328	0.397
MRR on Test	0.0012	0.021	0.0356	0.392

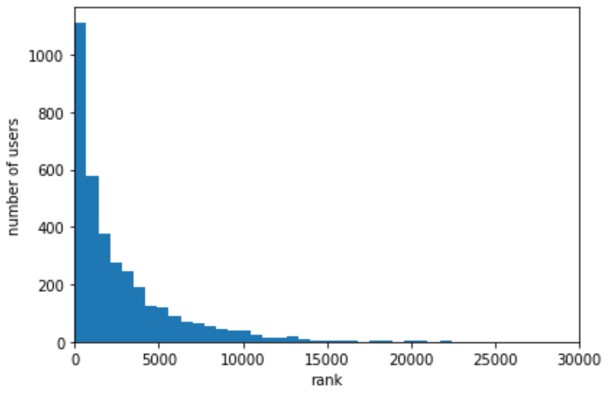


Figure 11: MRR by Exact Posts

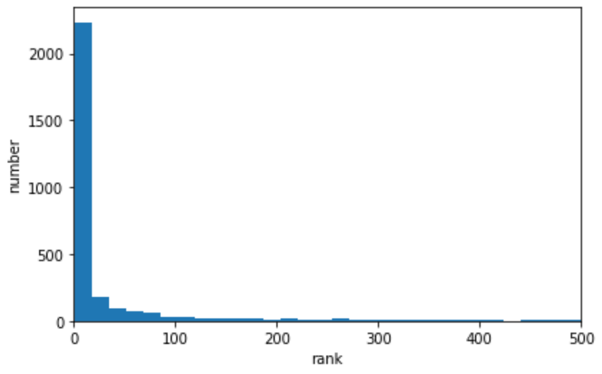


Figure 12: MRR by Similar Posts

We also evaluated our method with the whole dataset with different number of clusters to take a look at the impact of K. Table VII shows the results of both models on the test set. We can see that for the wide and deep model, the MRR had a increase from $K = 50$ to $K = 100$. Considering the trade off between performance and computational cost, $K = 100$ could be a better choice in this problem.

Table VII: Results for the Two Learning Models

	Cluster Number	MRR, Threshold = 0.95	Number of Ranks < 10	Training Time
Similarity Learning	50	0.0356	126	2h15min
	100	0.1111	1246	2h12min
	200	0.0743	1186	2h16min
	400	0.2068	1243	2h32min
Wide and Deep	50	0.3924	2112	7h03min
	100	0.4731	2261	8h38min
	200	0.4980	2225	12h02min
	400	0.2932	1356	19h37min

5.4 Impediments

Below we listed some major issues we encountered in this project, and how we dealt with them.

- **GCP memory issue dealing with large dataset.** The event log dataset covers events over 15 months and is more than 500GB. We hence need additional persistent disk on our virtual machine. We also encountered memory issues when running pyspark jobs in vm. We solved it by submitting our jobs to Cloud Dataproc and used multiple workers for faster job running.
- **User identification.** As mentioned before, digital trend website does not have a login system for each user, so there is no user id in the dataset for long-time user identification. Instead, we use the domain user id which ensures that multiple users will never share the same id.
- **Duplicate clicks.** Many users revisited a specific article page multiple times, leading to duplicate clicks in training set and test set. This actually increased our evaluation scores a lot. To make our evaluation more accurate, we removed those duplicate clicks.
- **No negative data points.** Since we don't have the information of articles listed on the same page at the same time. It is actually not possible for us to generate negative data points for training. We observed that the website itself tends to update the start page with the newest articles, so we assumed that articles published at the same time have larger possibilities to be listed at the same time. In this way, we generated our negative data point.
- **Jupyter Notebook Disconnection.** When performing heavy computation, our jupyter notebook often disconnected. So we used tmux on virtual machine to run our scripts remotely.
- **RAM limit dealing with re-clustering.** We attempted to re-cluster the posts whose sizes are above average, but 'MemoryError' always showed up for re-clustering involved intermediate steps to store large arrays, which causes total memory demands surpass our current RAM limit. We solved this problem by substituting the memory-demanding steps with manually joining the arrays with text editors.

6 Future Work

Our project only serves as an initial step for building a real recommendation system for Digital Trend website. There are a lot of future works to be done. To further improve the dataset, there are some information essential to collect. For example, the contents of each news article are needed for a better understanding and more precise clustering. Also, for each event session, we need to know all the listed news articles during each page request, this helps us collect negative data points, which is what a user did not click on and what were less likely to be clicked in the future. Also, we can try different learning models and further tuning the hyperparameters. For evaluation, since the website does not have recommendation system before, it makes more sense to perform an A&B test instead of offline evaluation to see if the one with personalized content can increase user's time of engagement with the website. Also, there are many important problems in recommendation systems that we did not consider in this project. For example, we did not deal with the cold start problem, as well as model updating when new data point arrives. Also, when making recommendations, articles' recency and big news event are also important factors to consider.

7 Brief Conclusion

In this project, we made use of the content history and event log data collected from digital trend website and tried to build a recommendation system for them. We used a similarity learning model as a baseline and compared the result with our wide and deep learning model. Offline experiment results show that our model is not sufficient to make the exact news article predictions. There are several explanations. The first is due to the limitation of our model. It may not be good at understanding users' behavior. Also, the data collected also has some limitations such as lack of content information and negative data points. However, results show that our model is good at predicting high similarity posts, which means giving recommendations that have similar keywords to users' interest. Our project serves as an initial step for building a real recommendation system for the website. We explored their dataset and found out some potentials and limitations. To continue the work, more future works need to be done as discussed in the previous section.

8 Acknowledgments

We would like to acknowledge our project mentor Rok Sosic and Jeffrey Ullman for guiding us on this project, and cs341 course teaching assistants Johannes Birgmeier and Abhay Agarwal for giving insightful input. We also thank the Digital Trend Website for providing us with the dataset, and their data scientists for giving suggestions through a bi-weekly meeting with us.

References

[1] Purnima Bholowalia and Arvind Kumar. Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105 09:0975–8887, 2014.

[2] Heng-Tze Cheng, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah, Levent Koc, Jeremiah Harmsen, and et al. Wide deep learning for recommender systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*, 2016.

[3] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271–280, New York, NY, USA, 2007. ACM.

[4] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems – survey and roads ahead. *Information Processing Management*, 54(6):1203 – 1227, 2018.

[5] Jiahui Liu, Elin Pedersen, and Peter Dolan. Personalized news recommendation based on click behavior. In *2010 International Conference on Intelligent User Interfaces*, 2010.