A Personalized
News Recommender System

IDC410 - Machine Learning

Project Report
01-05-2020

Contents

1 Introduction

We use machine learning to build a personalized news articles recommendation system based on the user's previous article ratings. Different people have different topics of interest, and this is not reflected in a single score. Our recommender system helps users instantly discover news articles to their liking, regardless of how distinct their tastes may be. If a new user lands on our app, we recommend news articles from diverse topics so that the probability of the user to find the article of his/her interest increases.

Current recommender systems generally fall into two categories: content-based filtering and collaborative filtering. We experiment with both approaches in our project. For content-based filtering, we use the TF-IDF (term frequency/inverse-document-frequency) technique for vectorization of data and further calculate the similarity between articles. We have also used a topic modeling based approach for the same. For collaborative filtering, the input to our algorithm is the observed users' article rating, and we use K-nearest neighbors and matrix factorization to predict user's article ratings.

2 Dataset and Attributes

News articles dataset - We created a corpus of ~10,000 news articles taken from multiple sources, the major contributors being the following: Scroll, HistoryNewsNetwork, The Verge, and ESPN. The data corpus contains multiple features of the scraped articles including their date of publication, article id, author name, article headline, article topic.

User Clickstream dataset - Once the user starts consuming news stories, (s)he leaves behind a clickstream which can be then used to uncover user interests and provide personalized recommendations. We generated a user clickstream data which contains the following attributes:

User ID	-	ID given to each unique user on the app
Session ID	-	Refers to each new visit of the user on the app
Article ID	-	A unique ID given to each article
Click	-	Takes the value 'Yes' if the article has been clicked by the user, otherwise no.
Time spent	-	Time spent by the user on a particular article in seconds
Rating	-	Rating of an article ranging from 1 to 5. Correlated with time spent.

A Naive Assumption ⇒ The users spending more time on a given article tend to like that article and give a higher rating, i.e. the time spent and rating are correlated variables.
We're ignoring the case in which there can be a user who spends much time reading an article and still dislikes it. We're assuming that since the user read the article for some time, he/she would at least be interested in the topic of that article.

The user clickstream data generated is of the following form:

User id	Session id	Article id	Click	Ratings	Time spent (sec)
49	1	23777	yes	3	103.64
39	1	20800	yes	-	76.34
93	1	20647	yes	2	27.57
69	1	24602	no	-	0.00
14	1	23716	yes	-	67.88
90	1	19747	yes	-	64.73
79	1	24464	yes	5	154.97

Figure 1: Typical user clickstream data

For generating the above dataset, we have assumed that our app encounters 3 types of users, i.e. The users who :

- **Land and bounce** - These are the users who do not show any activity on the news website. Their time spent value ~ 0 and 'click' value remains 'no'. We do not have any article ratings by these users.
- **Land, read, and never come back** - We have generated the correlated variables **'time spent'** and **'rating'** from a multivariate normal distribution. For this category of users, the number of articles read by the user in a single session is chosen randomly from 1 to 3 with weights of 0.7, 0.2, and 0.1 respectively.
- **Repeat users** - These users repeatedly visit the app and we're assuming the number of articles read by such a user in a single session takes values from 1 to 4 with weights of 0.4, 0.3, 0.2, and 0.1.

3 Implementation

3.1 The Cold Start Problem

The idea here is to cluster the articles into a few categories and recommend one article from each category so as to provide the new user a diverse set of options. We created a TF-IDF Vectorizer which is used to weigh a keyword in any document and assign importance to that keyword based on the number of times it appears in the document. This produced a matrix of shape (10239, 37233). Now, to how many dimensions to reduce our high-dimension matrix seemed difficult to answer. Sklearn's official recommendation states, 'For Latent Semantic Analysis (what we're doing here), a value of 100 is recommended.' So, SVD (Singular Value Decomposition) was then used to reduce the data to 100 dimensions followed by Kmeans clustering.

3.2 Content-based recommendation systems (CBRS)

These systems recommend items with similar content from the ones the user liked in the past. In this type of RS, the clickstream based user profile (up) represents the user interests. We have implemented the following two approaches for content-based system :

- **Tfidf based approach** - A tfidf_matrix containing each word and its TF-IDF score with regard to each document was generated and the cosine similarity of the whole corpus with user read articles was further calculated. Thus, each new item is compared to user profile, and the most similar articles are recommended. It minimizes the cold start problem: new items can be suggested before being rated by a substantial number of users as opposed to collaborative filtering.
- **Topic Modeling based approach** - We used the LDA (Latent Dirichlet Allocation) model on the cleaned/preprocessed news articles. The articles were represented in terms of Topic Vector and users in terms of the Topic Vector of read articles. We then calculated the similarity of cosines between read articles topic vector and total articles topic vector and give recommendations based on the same.

3.3 Collaborative recommendation systems

Collaborative recommendation systems use an approach that focuses on the relations between users or articles. The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion/taste as a person B on a set of items, A is more likely to have B's opinion for a given item than that of a randomly chosen person. Thus, the recommended articles are selected from users who have read similar articles based on their history. We cannot use collaborative recommenders in the beginning as there won't be significant user information to perform the recommendation.
We have used the following two approaches for user-based collaborative filtering:

- **Collaborative filtering based on user similarity score** - We created a users vs document rating matrix and use it to find the cosine similarity between different users. We estimate the unknown rating by taking weighted average and then recommend 10 articles based on the estimated ratings.

- **Collaborative filtering based on matrix decomposition/SVD** - We created the user vs article rating matrix and reduced its dimensions using Singular Value Decomposition method. The user vs article matrix was then re-calculated to recommend articles based on the estimated ratings.

4 Evaluation

4.1 Silhouette Score

For validation of the K-means clustering algorithm, we used the silhouette score to decide the optimum number of clusters to be formulated from the data. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.[1] For our system, we found the best silhouette score of +0.091 when the number of k-means clusters was taken to be 9. Though the highest peak is for around 81 clusters (on running the code for range 100), we still cluster it into 9 diverse clusters (which was a local peak instead) so as not to miss out on several categories.

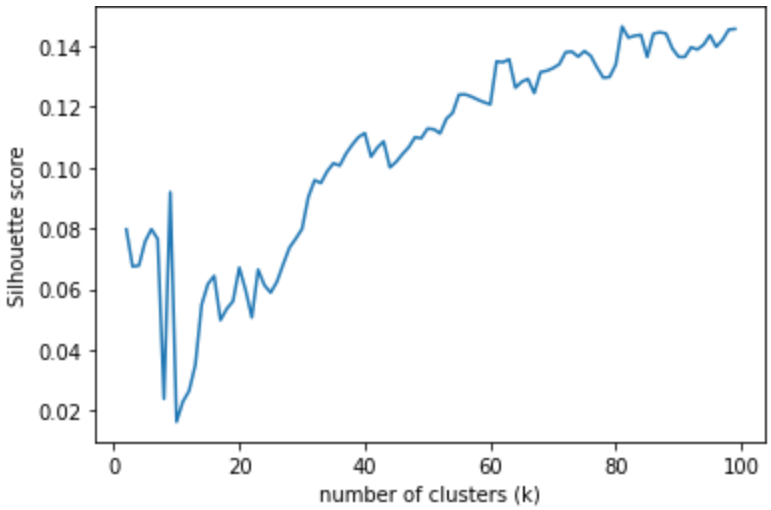

Figure 3: Silhouette score plot for k means clustering of articles

4.2 NDCG Score

We approach recommendation as a retrieval task, meaning that we're mainly interested in relatively few items that we consider most relevant and are going to show to the user. This is known as *the top-K* recommendation. The two most popular ranking metrics are MAP (Mean Average Precision) and NDCG (Normalized Discounted Cumulative Gain).

We have used the NDCG metric as it allows relevance scores in the form of real numbers while MAP assumes binary relevance (an item is either of interest or not). Intimidating as the name might be, the idea behind NDCG is pretty simple. A recommender returns some items and we'd like to compute how good the list is. Each item has a relevance score(based on article ratings in our case), usually a non-negative number. That's gain. The items for which we don't have user feedback for, we usually set the gain to zero. Now we add up those scores; that's

cumulative gain. We'd prefer to see the most relevant items at the top of the list, therefore before summing the scores we divide each by a growing number (usually a logarithm of the item position) - that's discounting - and get a DCG. DCGs are not directly comparable between users, so we normalize them.

S.No.	Model Name	NDCG Score
1.	TF-IDF based Recommendation System	0.727

We could not debug some errors in the evaluation of other models due to time constraint.

5 Inferences

- NDCG score looks very high for our TF-IDF based recommender. We believe that this happened because we ran it for very less data. Cross-validation methods can possibly improve this problem.
- Implicit methods of data collection can also be very helpful to learn user preferences.

6 Conclusions and Future Work

We have explored both content-based and collaborative filtering for building the recommendation system. The user profile information was constructed implicitly by tracking users' click actions so as to gather information in a non-intrusive manner. The major advantage of this implicit feedback method being that users do not need to provide any additional information about themselves.

Besides the results obtained with this work, there are many challenges to recommender systems that can be approached in future work or those that we could not address due to time constraints in the current system. In this section we propose some improvements that can be done to this developed system, namely:

1. **Multi-armed bandit solutions** to cold start problem - These reinforcement learning algorithms help in continuously balancing exploration with exploitation. It tries to reward the system whenever it goes right and reduces the regrets i.e to exploit the highest max payoff and still explore to get more information about the user preferences.[2] We didn't incorporate multi-armed bandit testing in our current system although it was our plan originally due to its computational complexity. It seemed more difficult and resource-intensive to run multi-armed bandit tests.

2. The system can be extended by trying different recommendations and **hybridization techniques** like feature augmentation hybridization or weighted hybridization based on which one works better for our textual domain.

3. We could not implement **Cross validation method** for the evaluation of our system due to time constraint but we would very much like to incorporate it to the system in future

and use more evaluation metrics like coverage and diversity to test our model.

4. Improvements can be done on the textual content analysis. The recommendation algorithm proposed here analyses the textual content of the news. This analysis consists of the computation of a text vectorizer through the tf-idf method. The more similar the textual content between two news, the more relevance will have in the prediction evaluation of each other. Nevertheless, a deeper analysis of the texts can contribute to better results. **Data mining techniques** can be introduced to identify the news that addresses the same subjects.

5. In the current representation of this system, users are selecting articles randomly from our recommended lists. We could incorporate **hard code user preferences** in our recommendation systems like saying some are interested in only technology and entertainment news while others can be more inclined to only political news. Such data could also be generated for giving a real world sense to the system.

7 References

1. Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". *Computational and Applied Mathematics*. 20: 53–65. doi:10.1016/0377-0427(87)90125-7.
2. Gittins, J. C. (1989), *Multi-armed bandit allocation indices*, Wiley-Interscience Series in Systems and Optimization., Chichester: John Wiley & Sons, Ltd., ISBN 978-0-471-92059-5