**Building a Collaborative Filtering Recommender System with ClickStream Data**

**How to implement a recommendation algorithm based on prior implicit feedback.**

**https://towardsdatascience.com/building-a-collaborative-filtering-recommender-system-with-clickstream-data-dffc86c8c65**

[Recommender systems](https://en.wikipedia.org/wiki/Recommender_system) are everywhere, helping you find everything from books to romantic dates, hotels to restaurants.

There are [all kinds of recommender systems for all sorts of situations](https://github.com/grahamjenson/list_of_recommender_systems), depends on your needs and available data.

**Explicit vs Implicit**

Let’s face it, **explicit** feedback is hard to collect as they require additional input from the users. The users give explicit feedback only when they choose to do so. As a result, most of the time, people don’t provide ratings at all (I myself totally guilty of this on Amazon!). Therefore, the amount of explicit data collected are extremely scarce.

On the other hand, **implicit** data is easy to collect in large quantities without any effort from the users. The goal is to convert user behavior into user preferences which indirectly reflect opinion through observing user behavior. For example, a user that bookmarked many articles by the same author probably likes that author.

**The Data**

Our goal today is to develop a recommender system with implicit data collection which is clickstream data, in our case.

It is very hard to find public available data for this project. I am using data from [Articles sharing and reading from CI&T DeskDrop](https://www.kaggle.com/gspmoreira/articles-sharing-reading-from-cit-deskdrop). [Deskdrop](https://deskdrop.co/) is an internal communications platform that allows companies employees to share relevant articles with their peers, and collaborate around them.

The data contains about 73k users interactions on more than 3k public articles shared in the platform, more importantly, it contains rich implicit feedback, different interaction types were logged, making it possible to infer the user’s level of interest in the articles.

And we will be using [Implicit Library](https://github.com/benfred/implicit/), a Fast Python Collaborative Filtering for Implicit Datasets, for our matrix factorization.

**Data Pre-processing**

* Remove columns that we do not need.
* Remove eventType == 'CONTENT REMOVED' from articles\_df.
* Merge interactions\_df with articles\_df.

implicit\_rec\_preprocess.py

This is the data set that will get us to start:



Table 1

This tells us what event type each person has with each content. There are many duplicated records and we will remove them shortly.

df['eventType'].value\_counts()

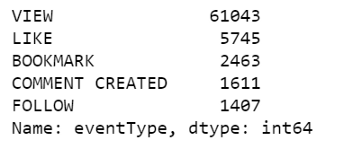


Figure 1

The eventType values are:

* **VIEW**: The user has opened the article. A page view in a content site can mean many things. It can mean that the user is interested, or maybe user is just lost or clicking randomly.
* **LIKE**: The user has liked the article.
* **BOOKMARK**: The user has bookmarked the article for easy return in the future. This is a strong indication that the user finds something of interest.
* **COMMENT CREATED**: The user left a comment on the article.
* **FOLLOW**: The user chose to be notified on any new comment about the article.

We are going to associate each eventType with a weight or strength. It is reasonable to assume that for example, a bookmark on an article indicates a higher interest of the user on that article than a like.

event\_type\_strength = {  
 'VIEW': 1.0,  
 'LIKE': 2.0,   
 'BOOKMARK': 3.0,   
 'FOLLOW': 4.0,  
 'COMMENT CREATED': 5.0,   
}df['eventStrength'] = df['eventType'].apply(lambda x: event\_type\_strength[x])



Table 2

* Drop duplicated records.
* Group eventStrength together with person and content.

df = df.drop\_duplicates()  
grouped\_df = df.groupby(['personId', 'contentId', 'title']).sum().reset\_index()  
grouped\_df.sample(10)

* We get the final result of grouped eventStrength.

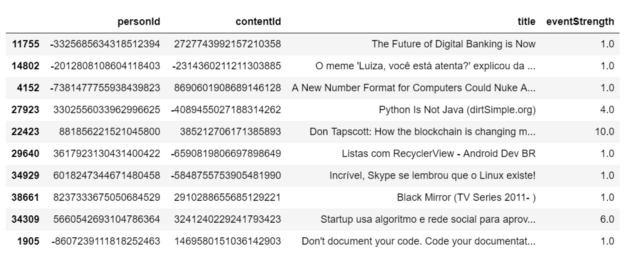


Table 3

**Alternating Least Squares Recommender Model Fitting**

Instead of representing an explicit rating, the eventStrength can represent a “confidence” in terms of how strong the interaction was. Articles with a larger number of eventStrength by a person can carry more weight in our ratings matrix of eventStrength.

* To get around “negative integer” warning, I will have to create numeric person\_id and content\_id columns.
* Create two matrices, one for fitting the model (content-person) and one for recommendations (person-content).
* Initialize the Alternating Least Squares (ALS) recommendation model.
* Fit the model using the sparse content-person matrix.
* We set the type of our matrix to double for the ALS function to run properly.

implicit\_als\_model.py

**Finding the Similar Articles**

We are going to find the top 10 most similar articles for content\_id = 450, titled “***Google’s fair use victory is good for open source***”, this article seems talk about Google and open source.

* Get the person and content vectors from our trained model.
* Calculate the vector norms.
* Calculate the similarity score.
* Get the top 10 contents.
* Create a list of content-score tuples of most similar articles with this article.

similar\_content.py

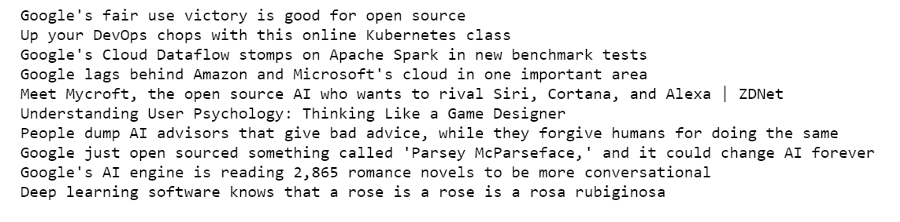


Figure 2

The 1st article is itself. The other 9 articles are about Google, or opensource software, or cloud, or AI, or the other tech companies. I am sure you will agree with me that they are all some what similar with the first one!

**Recommend Articles to Persons**

The following function will return the top 10 recommendations chosen based on the person / content vectors for contents never interacted with for any given person.

* Get the interactions score from the sparse person content matrix.
* Add 1 to everything, so that articles with no interaction yet become equal to 1.
* Make articles already interacted zero.
* Get dot product of person vector and all content vectors.
* Scale this recommendation vector between 0 and 1.
* Content already interacted have their recommendation multiplied by zero.
* Sort the indices of the content into order of best recommendations.
* Start empty list to store titles and scores.
* Append titles and scores to the list.
* Get the trained person and content vectors. We convert them to csr matrices.
* Create recommendations for person with id 50.

implicit\_rec\_als\_id\_50.py

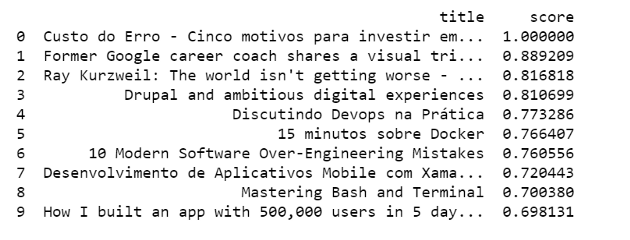


Figure 3

Here we have top 10 recommendations for person\_id = 50. Do they make sense? Let’s get top 10 articles this person has interacted with.

grouped\_df.loc[grouped\_df['person\_id'] == 50].sort\_values(by=['eventStrength'], ascending=False)[['title', 'person\_id', 'eventStrength']].head(10)

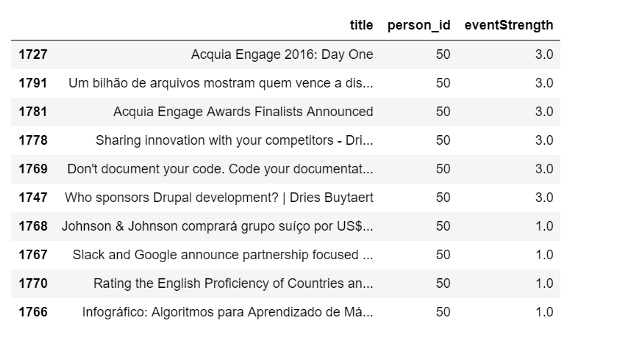


Table 4

Apparently, this person is interested in articles on open source CMS such as Drupal, she also reads software development and business related articles, namely “Google”, “Slack” or “Johnson Johnson”.

The articles we recommended to her includes Drupal for digital experience, information technology vs. humanity, software development, and business articles about Google.

Pretty impressive! Let’s try one more.

We recommended the following articles to person\_id = 1:

person\_id = 1recommendations = recommend(person\_id, sparse\_person\_content, person\_vecs, content\_vecs)print(recommendations)

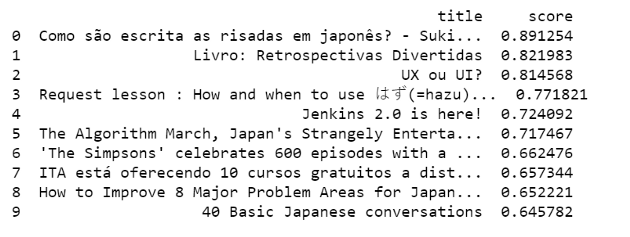


Figure 4

The following are the articles person\_id = 1 has interacted with:

grouped\_df.loc[grouped\_df['person\_id'] == 1].sort\_values(by=['eventStrength'], ascending=False)[['title', 'eventStrength', 'person\_id']]

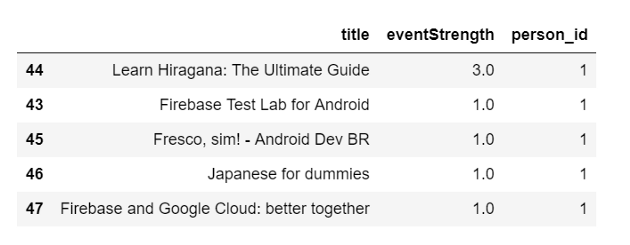


Table 5

Apparently, this person has only interacted with 5 articles and she seems to have very limited interest. The articles she interacted was about learning Japanese language and / or android development.

The articles we recommended to her includes learning Japanese language, android development and user interface design. Cool!

**Evaluation the Recommender System**

The above spot checks look all good. But the best evaluation metrics for a recommender system is how much the system adds value to the end user and/or business, whether the system increase page views, likes, bookmarks, follows and comments. We will want to do some kind online A/B testing to evaluate these metrics.

However, before we push the recommender system online, there are other common metrics for evaluating the performance of a recommender in isolation. By following this [tutorial](https://nbviewer.jupyter.org/github/jmsteinw/Notebooks/blob/master/RecEngine_NB.ipynb), we were able to calculate the [AUC](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) for each person in our training set that had at least one article masked. And [AUC](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) for the most popular articles for the persons to compare.



[Jupyter notebook](https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Articles%20Rec%20System%20Implicit.ipynb) can be found on [Github](https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Articles%20Rec%20System%20Implicit.ipynb). Happy Easter!

References:

**[AlternatingLeastSquares - Implicit 0.3.8 documentation](https://implicit.readthedocs.io/en/latest/als.html?source=post_page---------------------------)**

**[A Recommendation Model based off the algorithms described in the paper 'Collaborative Filtering for Implicit Feedback…](https://implicit.readthedocs.io/en/latest/als.html?source=post_page---------------------------)**

**[implicit.readthedocs.io](https://implicit.readthedocs.io/en/latest/als.html?source=post_page---------------------------)**

**[ALS Implicit Collaborative Filtering](https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe?source=post_page---------------------------)**

**[Continuing on the collaborative filtering theme from my collaborative filtering with binary data example i’m going to…](https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe?source=post_page---------------------------)**

**[medium.com](https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe?source=post_page---------------------------)**

**[Recommender Systems in Python 101](https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101?source=post_page---------------------------)**

**[Using data from Articles sharing and reading from CI&T DeskDrop](https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101?source=post_page---------------------------)**

**[www.kaggle.com](https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101?source=post_page---------------------------)**

**[Towards Data Science](https://towardsdatascience.com/?source=post_sidebar--------------------------post_sidebar-)**

**Sharing concepts, ideas, and codes.**

Follow

**326**

* [Machine Learning](https://towardsdatascience.com/tag/machine-learning)
* [Recommendation System](https://towardsdatascience.com/tag/recommendation-system)
* [Python](https://towardsdatascience.com/tag/python)
* [Artificial Intelligence](https://towardsdatascience.com/tag/artificial-intelligence)
* [Implicit](https://towardsdatascience.com/tag/implicit)

**326 claps**

[](https://towardsdatascience.com/@actsusanli?source=follow_footer--------------------------follow_footer-)

Written by

[**Susan Li**](https://towardsdatascience.com/@actsusanli?source=follow_footer--------------------------follow_footer-)

Follow

**Changing the world, one post at a time. Sr Data Scientist, Toronto Canada. https://www.linkedin.com/in/susanli/**